

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

МЕТОДИЧНІ ВКАЗІВКИ
до лабораторних занять та самостійної роботи
**«МЕТОДИ КОНСТРУЮВАННЯ ОБ'ЄКТІВ
В КОМП'ЮТЕРНИХ СИСТЕМАХ»**
(Частина IV)

для студентів спеціальності 122
«Комп'ютерні науки»

Затверджено
редакційно-видавничою
радою університету,
протокол № 2 від 24.05.2018 р.

Харків 2019

Методичні вказівки для студентів спеціальності 122 «Комп'ютерні науки».
Методи конструювання об'єктів в комп'ютерних системах. Частина IV / укл.
І. Ю. Адашевська. – Х.: НТУ «ХПІ», 2019. – 56 с.

Рецензент: Л.М.Савченко.

Кафедра геометричного моделювання та комп'ютерної графіки

Частина 1. ІНФОРМАЦІЙНІ СИСТЕМИ ІМІТАЦІЙНОГО МОДЕЛЮВАННЯ

Імітаційне моделювання (ситуаційне моделювання) – метод, що дозволяє будувати моделі, які описують процеси так, як вони повинні відбуватися у дійсності. Таку модель можна «програти» у часі як для одного випробування, так і для заданої кількості. При цьому результати будемо визначати за випадковим характером процесів. За цими даними можна отримати достатньо стійку статистику.

Імітаційне моделювання – це метод дослідження, при якому систему, що вивчають, можливо замінити моделлю, яка з достатньою точністю описує реальну систему, з якою проводять експерименти з метою отримання інформації про цю систему. Експериментування з моделлю називають імітацією (імітація – це розуміння суті явища, не вдаючись до експериментів на реальному об’єкті).

Імітаційне моделювання – це окремий випадок математичного моделювання. Існує клас об’єктів, для яких з різних причин не розроблені аналітичні моделі, або не розроблені методи вирішення отриманої моделі. У цьому випадку аналітичну модель замінюють імітатором або імітаційною моделлю.

Імітаційним моделюванням іноді називають отримання приватних чисельних рішень сформульованої задачі на основі аналітичних рішень або за допомогою чисельних методів.

Імітаційна модель – логіко-математичний опис об’єкта, який може бути використано для експериментування на комп’ютері з ціллю проектування, аналізу та оцінки функціонування об’єкта.

Імітаційне моделювання застосовують у ситуаціях, коли:

- дорого або неможливо експериментувати на реальному об’єкті;
- неможливо побудувати аналітичну модель: в системі є час, причинні зв’язки, наслідок, нелінійності, стохастичні (випадкові) змінні;
- необхідно зімітувати поведінку системи за часом.

Мета імітаційного моделювання полягає у відтворенні поведінки досліджуваної системи на основі результатів аналізу найбільш суттєвих взаємозв’язків між її елементами або іншими словами – розробці симулятора (англ. **Simulation modeling**) досліджуваної предметної області для проведення різних експериментів.

Лабораторна робота №1

Тема: конструювання об'єктів із застосуванням імітаційної системи моделювання.

Мета роботи: реалізувати в середовищі *AnyLogic* модель банкомату, в рамках якого забезпечити можливість: завантажувати банківську карту в термінал, переглядати баланс на рахунку, реалізувати процес вилучення карти з терміналу, імітацію миготіння вікна прийому банківських карт.

Теоретичні відомості

1. Інтерфейс програми

Запускаємо AnyLogic, відкривається початкова сторінка, яка має стислий опис основних можливостей програми, посилання на приклади моделей, що поставляють разом з AnyLogic, а також посилання на веб-сайт компанії **XJ Technologies** (розробника цього програмного продукту) і на форму зворотного зв'язку з компанією.

Для того, щоб закрити початкову сторінку, клацаємо на кнопці («X»)



в панелі заголовка початкової сторінки.

За допомогою кнопки **Відкрити модель** панелі інструментів або команди **Файл / Відкрити** в головному меню вибираємо файл **Balls**. Це нескладна модель м'яча, який стрибає. На екрані Ви побачите вікно – рис. 1.1.

AnyLogic під час відкриття проекту відкриває кілька панелей: проекти, діаграма, палітра, помилки і властивості. Рис. 1 показує основні складові призначеного для користувача інтерфейсу. Розглянемо їх по черзі.

Панель **Проекти** забезпечує навігацію за елементами відкритих моделей. Оскільки модель має ієрархічну організацію, то вона відображена у вигляді дерева: сама модель утворює верхній рівень дерева; класи активних об'єктів і експерименти утворюють наступний рівень і т.д.

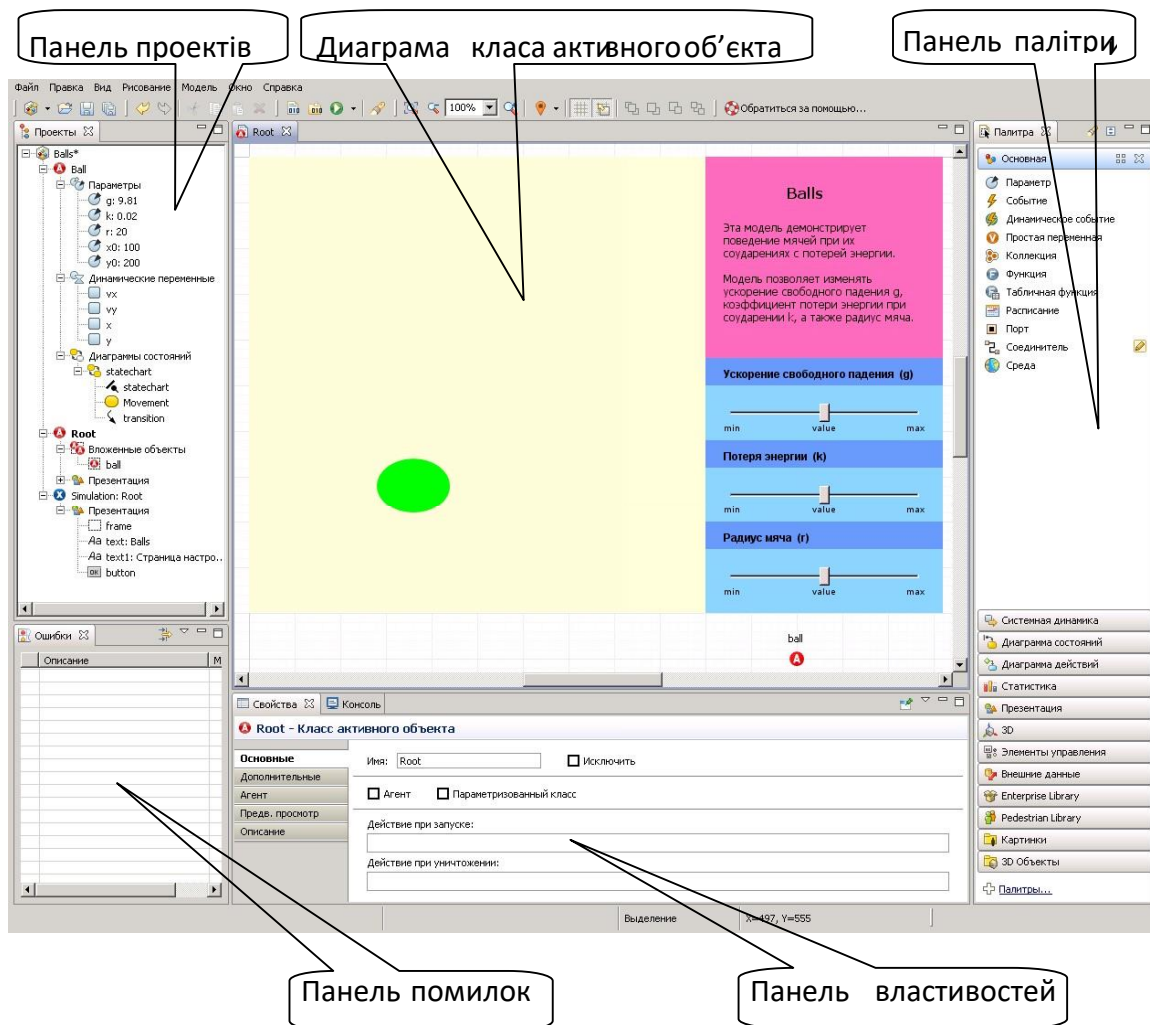


Рисунок 1.1 – Основні складові призначеного для користувача інтерфейсу

Панель **Проекти** за замовчуванням прикріплено до лівої частини робочої області *AnyLogic*.

Напівжирним шрифтом на дереві виділяємо той елемент, редактор якого активний на даний момент.

У випадку внесення до моделі будь-яких змін і їх видалення, модель буде відразу ж виділено на дереві – до імені моделі буде додано зірочку (*).

Можна розгортати і згорнути гілки дерева елементів моделі за допомогою кнопок + і -.

Кожний клас активного об'єкта і експерименту має свою діаграму, яку можна редагувати в графічному редакторі.

На діаграмах можна відтворювати такі дії:

- Малювати презентацію за допомогою фігур і елементів управління.
- Задавати поведінку активного об'єкта за допомогою подій і діаграм дій.
- Задавати структуру класу, додавши вкладені об'єкти.

- Додавати до презентації візуалізуючі графіки, діаграми.

Панель **Палітра** містить елементи, які можна додати на діаграму класу активного об'єкта або експеримента.

За умовчанням її прикріплено до правого краю вікна програми.

Панель Палітри складається з декількох вкладок (палітр), кожна з яких містить елементи, які стосуються певної задачі:

- **Основна** містить основні елементи, за допомогою яких можна задати динаміку моделі, її структуру і дані.
- **Системна динаміка** містить: елементи діаграми потоків і накопичувачів, а також параметр, з'єднувач і табличну функцію.
- **Діаграма станів** містить блоки діаграм, що дозволяють графічно задавати поведінку об'єкта.
- **Діаграма дій** містить блоки структурованих блок-схем, що дозволяють задавати алгоритми візуально.
- **Статистика** містить елементи, які використовують для збору, аналізу і відображення результатів моделювання.
- **Презентація** містить елементи для малювання презентацій: примітивні фігури, а також елементи управління, з метою надання презентації інтерактивності.
- **Зовнішні дані** містить інструменти для роботи з базами даних і текстовими файлами.
- **Зображення** містить набір картинок найбільш часто модельованих об'єктів: людина, вантажівка, фура, навантажувач, склад, завод і т. д.

Панель **Властивості** використовують під час перегляду і зміни властивостей обраного в даний момент елемента моделі.

Панель **Властивості** має кілька вкладок. У кожній вкладки є елементи управління, такі як поля введення, прапорці, перемикачі, кнопки і т.д., за допомогою яких можливо переглядати і змінювати властивості елементів моделі. Кількість вкладок і їх зовнішній вигляд залежить від типу обраного елемента.

На етапі компіляції моделі AnyLogic робить перевірку синтаксису діаграм, типів і параметрів. Всі виявлені на етапі компіляції і побудови моделі помилки мають відображення на панелі **Помилки**.

Кожна помилка має відображення опису і місця розташування – ім'я елемента моделі під час завдання якого ця помилка виникла.

Активний об'єкт є основним структурним елементом моделі в AnyLogic. Активним об'єктом називають сутність, яка містить дані, функції та поведінку як одне ціле. Активний об'єкт будують як клас, який може включати в якості складових елементів екземпляри інших класів активних об'єктів.

Наш проект **Balls** має два класи активних об'єктів: клас **Ball** і клас **Root**. На дереві проекту (рис.1.2), як складові елементи класу **Ball** показані **Параметри**, **Динамічні змінні** та **Діаграми станів**, у класі **Root** складовими його елементами показані **Вкладені об'єкти** і **Презентація**.

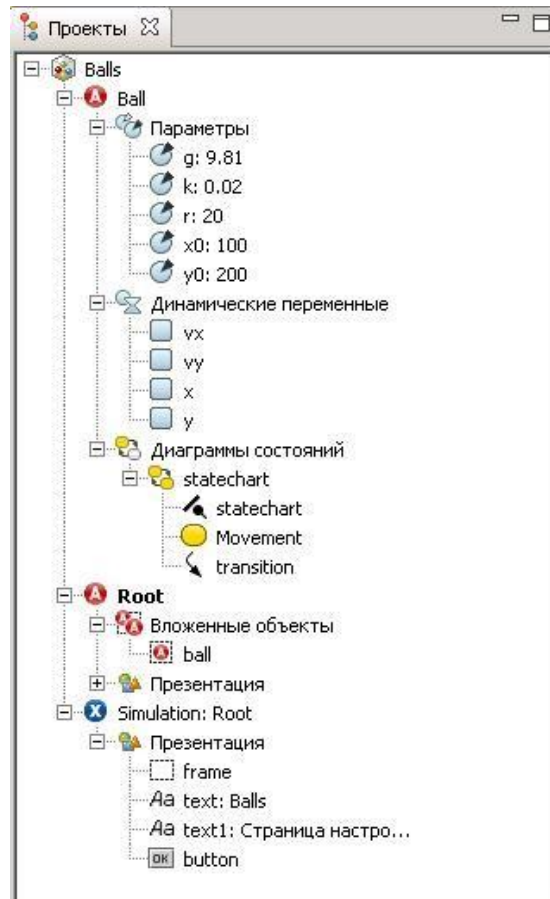


Рисунок 1.2 – Дерево проекту

Одна з гілок в дереві проекту має назву **Simulation**. Це експеримент, який може бути виконано з моделлю. За допомогою експериментів задають конфігураційні налаштування моделі. **AnyLogic** підтримує кілька типів експериментів, кожен з яких відповідає своєму завданню моделювання. AnyLogic підтримує такі:

- Простий експеримент
- Варіювання параметрів

- Оптимізація
- Порівняння «прогонів»
- Монте Карло
- Аналіз чутливості
- Калібрування
- Нестандартний

Можливо будь-як переміщувати панелі в межах вікна AnyLogic. Для відновлення прийнятих за замовчуванням налаштувань розташування панелей, потрібно в головному меню викликати **Вікно / Відновити розташування панелей**.

2. Діаграма класу активного об'єкта

У нашій моделі діаграму класу активного об'єкта – м'яча задають у вікні з ім'ям **Ball**, яке містить його змінні (координати м'яча x , y і його швидкості V_x і V_y), параметри (g , r , k , x_0 і y_0) і діаграма станів з ім'ям statechart (рис.1. 3). AnyLogic відображає отримані залежності між змінними за допомогою вузьких блакитних стрілок.

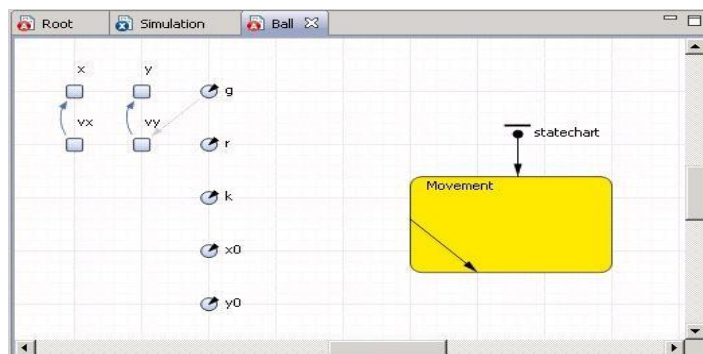


Рисунок 1.3 – Діаграма класу активного об'єкта

Стрілка, спрямована від змінної V_x до змінної x означає, що змінна V_x згадується у формулі змінної x . Стрілки залежностей малюються автоматично, завжди синхронізовані з формулами змінних і автоматично з'являються або зникають на діаграмі, як тільки змінна з'явиться у формулі або буде виключена. Для поліпшення наочності можна редагувати зовнішній вигляд стрілок залежностей, а саме змінювати їх колір і радіус заокруглення. Структура кореневого активного об'єкта **Root** задана в вікні з ім'ям **Root**. У моделі (рис. 1.1) активний об'єкт **Root** містить іконку з ім'ям ball – один екземпляр активного об'єкта Ball і анімацію.

3. Панель властивостей об'єктів моделі

Кожен елемент моделі має ті чи інші властивості або параметри. При виділенні будь-якого елемента в панелі **Проекту** або **Діаграми** внизу з'являється вікно властивостей саме цього виділеного елемента. Вікно **Властивості** (рис. 1.4) використовують для перегляда та зміни властивостей елементів.

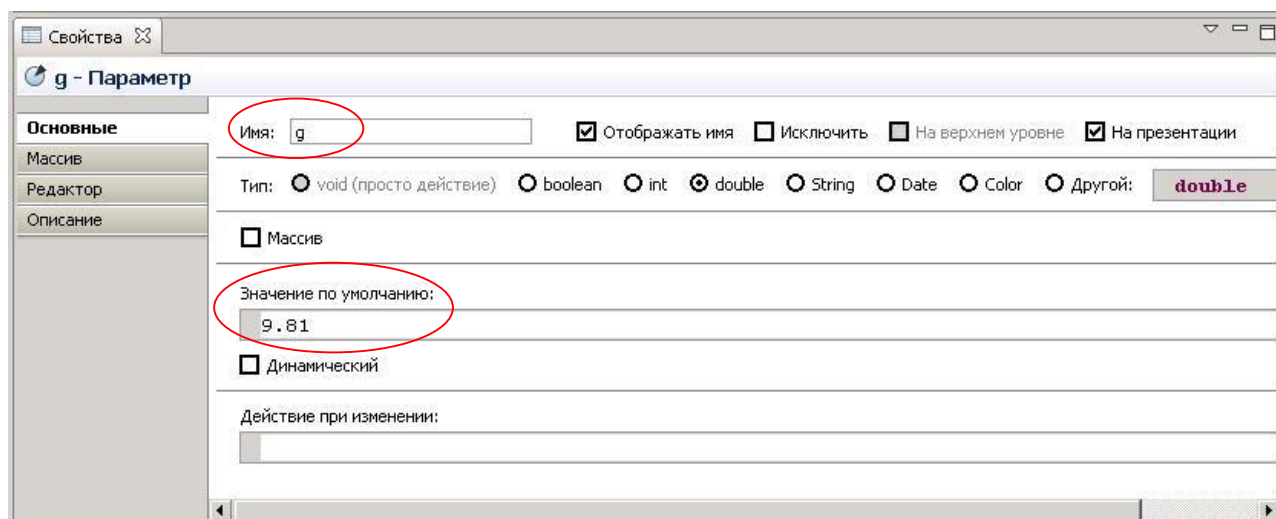


Рисунок 1.4 – Вікно **Властивості**

Наприклад, при виборі параметра *g* у вікні редактора діаграми об'єкта *Ball* внизу в панелі властивостей з'являться 4 вкладки: **Загальні**, **Масив**, **Редактор** і **Опис**. На вкладці **Загальні**, окрім імені цього об'єкта, вказані його параметри: тип (*double*), значення за замовчуванням (9.81) і відзначено, що слід показувати цей параметр на презентації та відображати його ім'я на діаграмі об'єкта *Ball*.

4. Поведінка активного об'єкта

«Поведінку» м'яча представлено в стейтчарті, рис. 1.3, який складається з **Початку діаграми станів**, одного стану з ім'ям *Movement* і одного переходу. Перехід спрацьовує при виконанні умови торкання поверхні землі при русі м'яча вниз, яке можна записати виразом:

$$y \leq r \ \&\& \ V_y < 0$$

Коли вертикальна координата *y* центра м'яча з радіусом *r* буде відстояти від поверхні на *r* і при цьому швидкість м'яча *V_y* буде спрямована донизу, перехід стейтчарта спрацює.

Цей вислів записано в полі **При виконанні умови** вікна властивостей переходу (рис. 1.5). При виконанні цієї умови м'яч відскакує, тобто його швидкість

змінює свій знак на протилежний і зменшується на долю k , що моделює втрату енергії при відскокуванні. Це відображено в полі **Дія** вікна властивостей переходу.

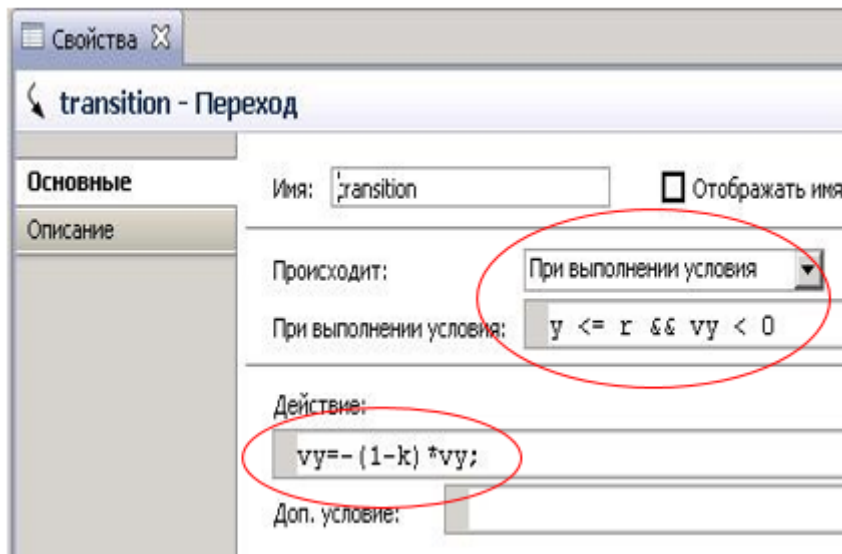


Рисунок.1.5 – Поле **При виконанні умови**

5. Презентація

У нашій моделі будемо двовимірне динамічне представлення, яке показує, що відбувається з моделлю з плином часу. Для моделі Balls в діаграмі Root побудовано зображення м'яча у вигляді зафарбованої окружності.

Презентація дозволяє більш наочно уявити динаміку модельованої системи, тобто її поведінку за часом для презентації геометричних фігур, наприклад окружності, що моделюють.

На рис. 1.6 можна побачити, що у вікні властивостей зеленого кола координати X і Y його центру і радіус r мають динамічні значення, які пов'язані зі змінними x , y і r примірника ball класу активного об'єкта Ball. Таким чином, зміна даних змінних буде викликати переміщення центру і зміну радіуса кола, що моделює м'яч, під час роботи моделі.

Клацнемо мишею на різних анімаційних об'єктах в панелі діаграми Root. Побачимо, що для різних елементів моделі вікно властивостей містить різні параметри, що характеризують саме цей елемент. Наприклад, клацнемо в презентації на слайдері. У вікні властивостей на вкладці **Основні** буде представлено інформацію про зв'язок слайдера з конкретним параметром моделі і геометричними параметрами самого слайдера.

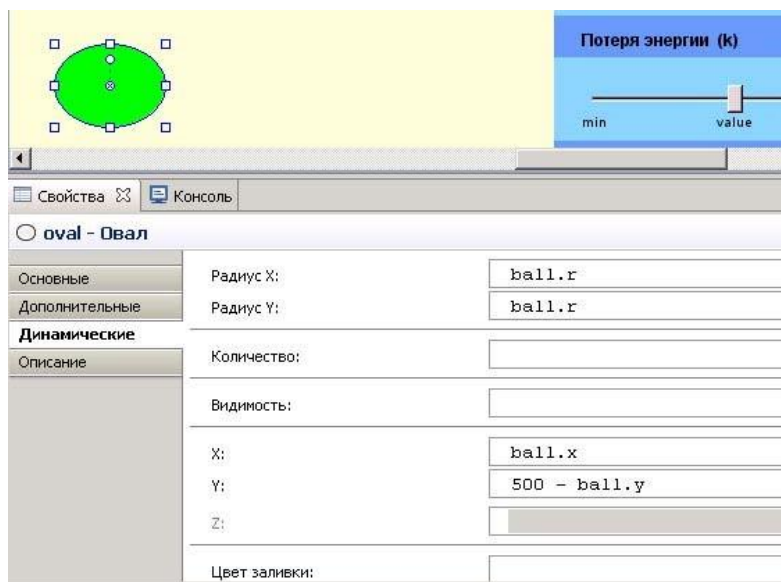


Рисунок. 1.6

Зверніть увагу, що в якості координати Y беремо значення змінної y зі знаком мінус і зміщенням 500 одиниць. Це зроблено тому, що вісь Y на презентації спрямована донизу, а не вгору, і знак мінус дозволяє нам перевернути вісь Y і спустити її до потрібного рівня.

Отже, модель Balls побудована і готова до запуску.

6. Режим виконання моделі

При запуску моделі можна виконувати різні експерименти з моделлю. Розглянемо основні засоби управління експериментом.

7. Запускання моделі

При відсутності помилок відкривається вікно презентації експерименту, рис.1.7, яке містить кнопку **Запустити модель і відкрити презентацію класу Main**.

При запусканні моделі за допомогою цієї кнопки, відкривається вікно презентації або експерименту, або одного з активних об'єктів запущеної моделі, рис. 1.8. На презентації буде видно всі елементи, у властивостях яких були встановлені прапорці **На презентації**.

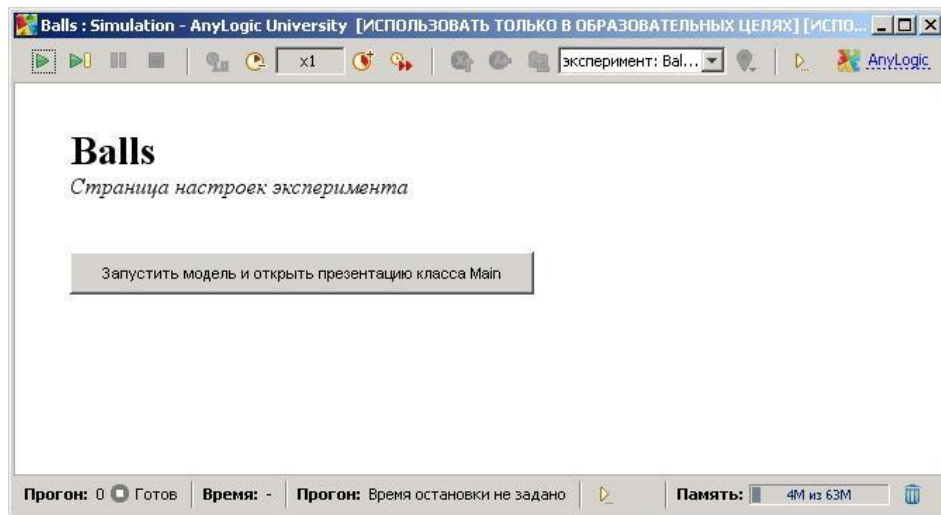


Рисунок 1.7 – Вікно презентації експерименту

При проведенні комп'ютерних експериментів можна використовувати всі кнопки, показані у верхній частині вікна (рис. 1.8):

- запуск або продовження моделювання;
- запуск виконання моделі по кроках ► ;
- пауза ■■;
- зупинка моделі і повернення у вікно презентації експерименту ■.

У нижній частині вікна можна побачити статус моделі (пауза або виконання, № прогону та інше, Інформація).

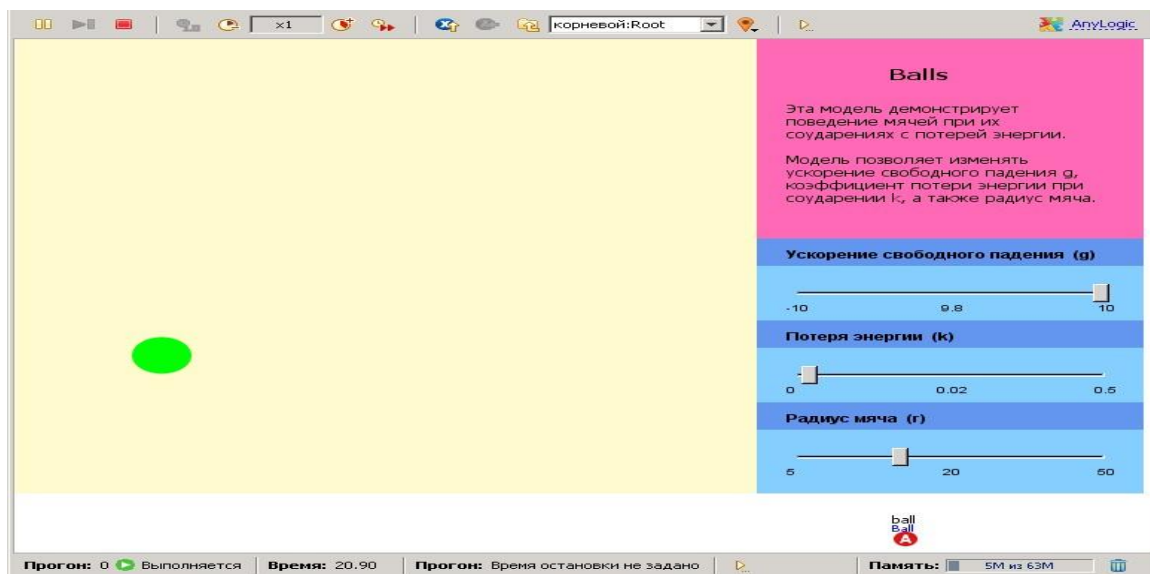


Рисунок 1.8

8. Експерименти з моделлю

На рис. 1.8. окрім рухомого зображення м'яча можна побачити текстовий коментар і «бігунки» або «слайдери» – рухливі покажчики для зміни параметрів моделі під час її виконання. Переміщуючи бігунки слайдерів, можна змінювати три параметра – прискорення вільного падіння, частку втрати швидкості м'яча під час кожного відскакування і радіус м'яча. Зміна параметрів дозволяє вивчати поведінку моделі в різних умовах – це і є комп'ютерний експеримент.




Проведіть кілька експериментів з моделлю, змінюючи параметри моделі.


9. Управління швидкістю виконання моделі та зображенням

Модель AnyLogic можна виконувати або в режимі віртуального, або в режимі реального часу. В режимі **віртуального часу** модель виконують без прив'язування до фізичного часу

– її просто виконують настільки швидко, наскільки це можливо. Цей режим найкраще підходить у тому випадку, коли потрібно моделювати роботу системи протягом досить тривалого періоду часу.

У режимі реального часу задають зв'язок модельного часу з фізичним, тобто задають кількість одиниць модельного часу, виконуваних за одну секунду. Це часто потрібно, коли необхідно, щоб відображення презентації моделі відбувалось з тією ж швидкістю, що і в реальному житті.

Перемикання між віртуальним і реальним часом виконання моделі здійснюють кнопкою **Віртуальний / реальний час**  панелі управління вікна презентації, а зменшення масштабу часу виконують за допомогою двох кнопок **Сповільнити** , **Прискорити**  і розташованого між ними поля.

Виконайте кілька експериментів з різними швидкостями виконання моделі, використовуючи кнопки керування. 

10. Навігація по моделі

Розташований в панелі управління вікна презентації, список що випадає **Навігація** відкриває організований у вигляді дерева список об'єктів моделі, забезпечуючи просту навігацію по моделі і швидкий доступ до будь-яких її об'єктів, рис.1.9. Коренем дерева об'єктів є кореневий об'єкт розпочатого експерименту. Якщо структура моделі змінюється під час виконання моделі, то ці зміни миттєво мають своє відображення в дереві об'єктів моделі.



Рисунок 1.9 – Список об'єктів моделі

Якщо вибрати об'єкт Ball, то ми побачимо його структурну діаграму зі значеннями змінних Vx і Vy , які динамічно змінюються. AnyLogic підтримує різні інструменти для збору, відображення та аналізу даних під час виконання моделі. Найпростішим способом перегляду поточного налаштування та історії зміни значень змінної або параметра під час виконання моделі є використання вікна **ІНСПЕКТ**.

Клацаємо мишею по значку змінної у вікні презентації. Побачимо невелике жовте вікно – це і буде вікно **інспекта**, рис.1.10.

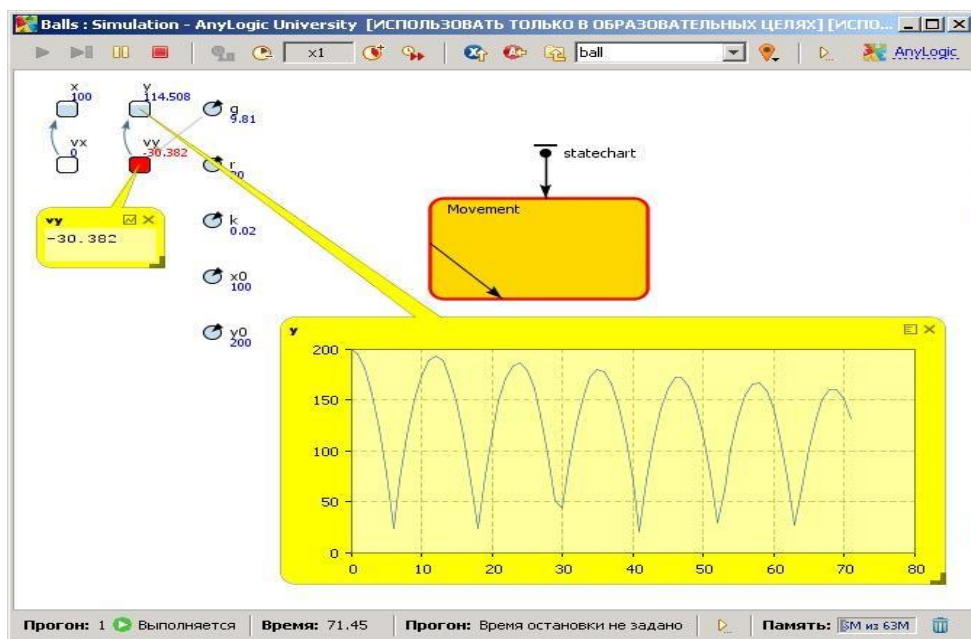


Рисунок 1.10 – Вікно **інспекта**

Встановимо відповідний розмір вікна шляхом перетягування нижнього правого кута вікна ІНСПЕКТ. Якщо потрібно, переміщуємо вікно, перетягуючи його мишею за панель назви вікна.

Порядок виконання роботи.

1. Для створення нової моделі необхідно виконати наступні дії: **Файл->Створити->Модель**. Перед вами з'явиться вікно **Нова модель**, в якому задаємо **Ім'я моделі**, **Розташування**, і назву **Java пакету**. Після натискання кнопки **Далі**, можна вибрати два варіанти: **Почати створення нової моделі з «нуля»**, або використовувати вже існуючий шаблон. Для створення моделі банкомату необхідно вибрати перше.

Після натискання кнопки **Готово**, система визначить початкову побудову моделі («чистий» лист) (рис.1.11).

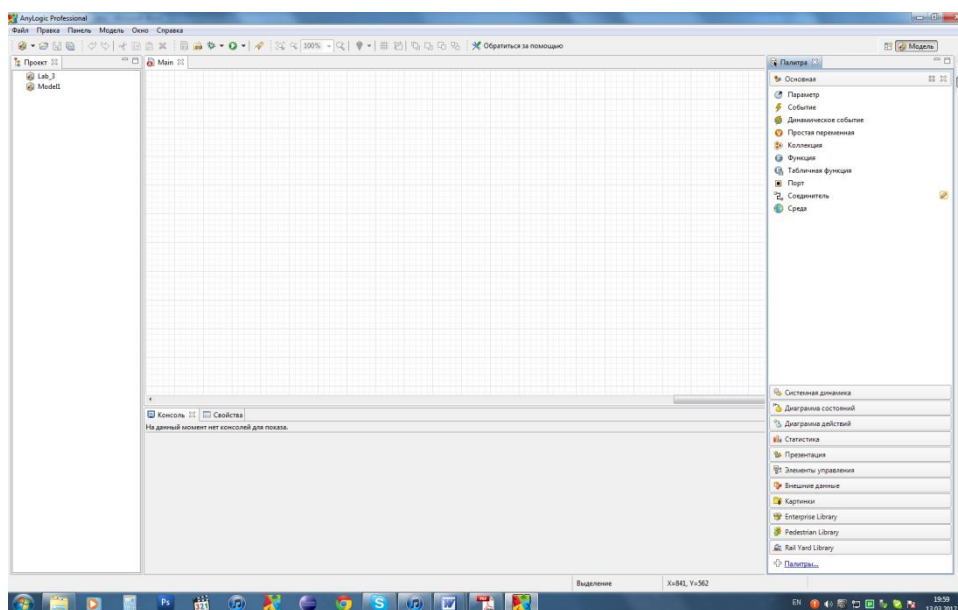


Рисунок 1.11 – Интерфейс системы моделирования AnyLogic в условиях порожней модели

2. Першим кроком буде необхідно додати картинку банкомату. Для цього необхідно в **Палітрі** вибрати вкладку **Презентація**, а з неї «перетягнути» на робочу область об'єкт **Зображення**. Щоб об'єкт відображав потрібне зображення, необхідно в його основних властивостях натиснути кнопку **Додати** і вибрати будь-яку «картинку» (рис. 1.12).

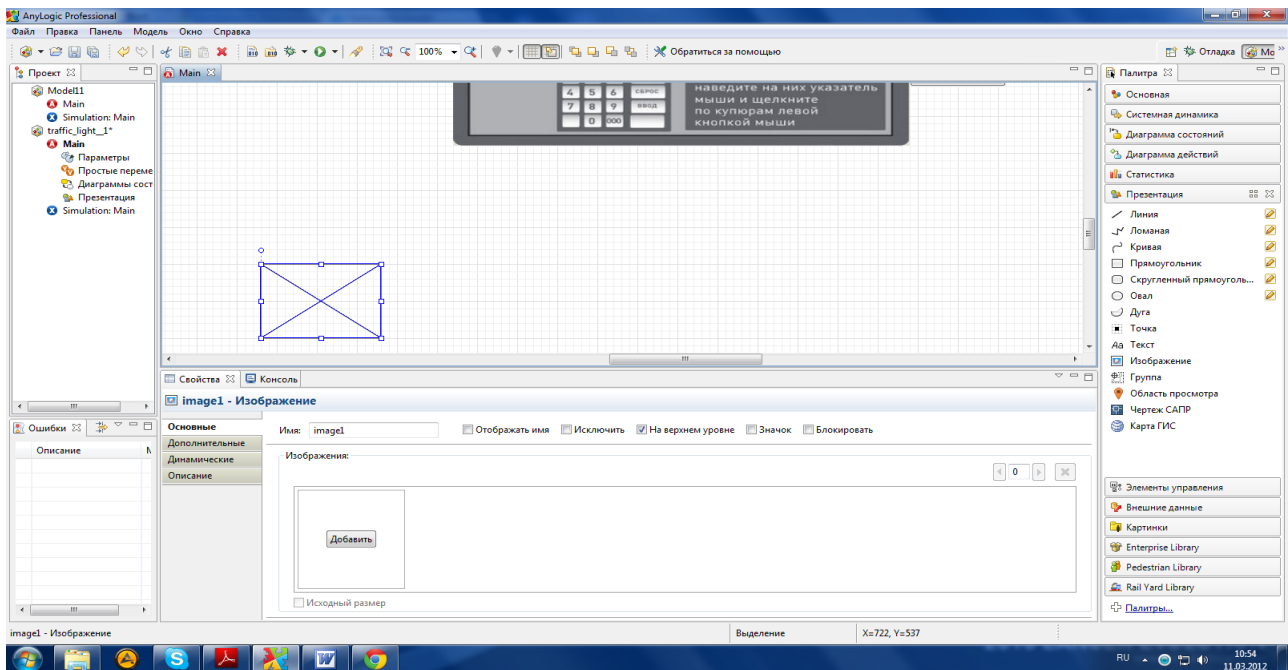


Рисунок 1.12 – Порожня модель AnyLogic з активною кнопкою «додати»

3. Наступний крок визначає створення імітації миготіння вікна прийому карток. Для цього перетягнемо з **Палітри** з вкладки **Основна** об'єкт **Параметр**, в основних властивостях якого визначимо наступні значення:

Ім'я – Card;

Тип – boolean.

Зверніть увагу, що після зміни властивості **Ім'я**, необхідно натиснути Ctrl + Enter. Інші параметри залиште без змін.

Додайте ще один **Параметр**, і визначте властивості:

Ім'я – insert_card;

Тип – boolean.

Ім'я – card;

Тип – boolean.

4. Далі на діаграмі класу активного об'єкта Model розмістимо **Початок діаграми станів** з панелі **Діаграма станів**. Відзначимо, що AnyLogic може працювати з елементами, набраними кирилицею. Перетягніть мишею елемент **Стан** під стрілочку початку діаграми як показано на рис. 1.13.

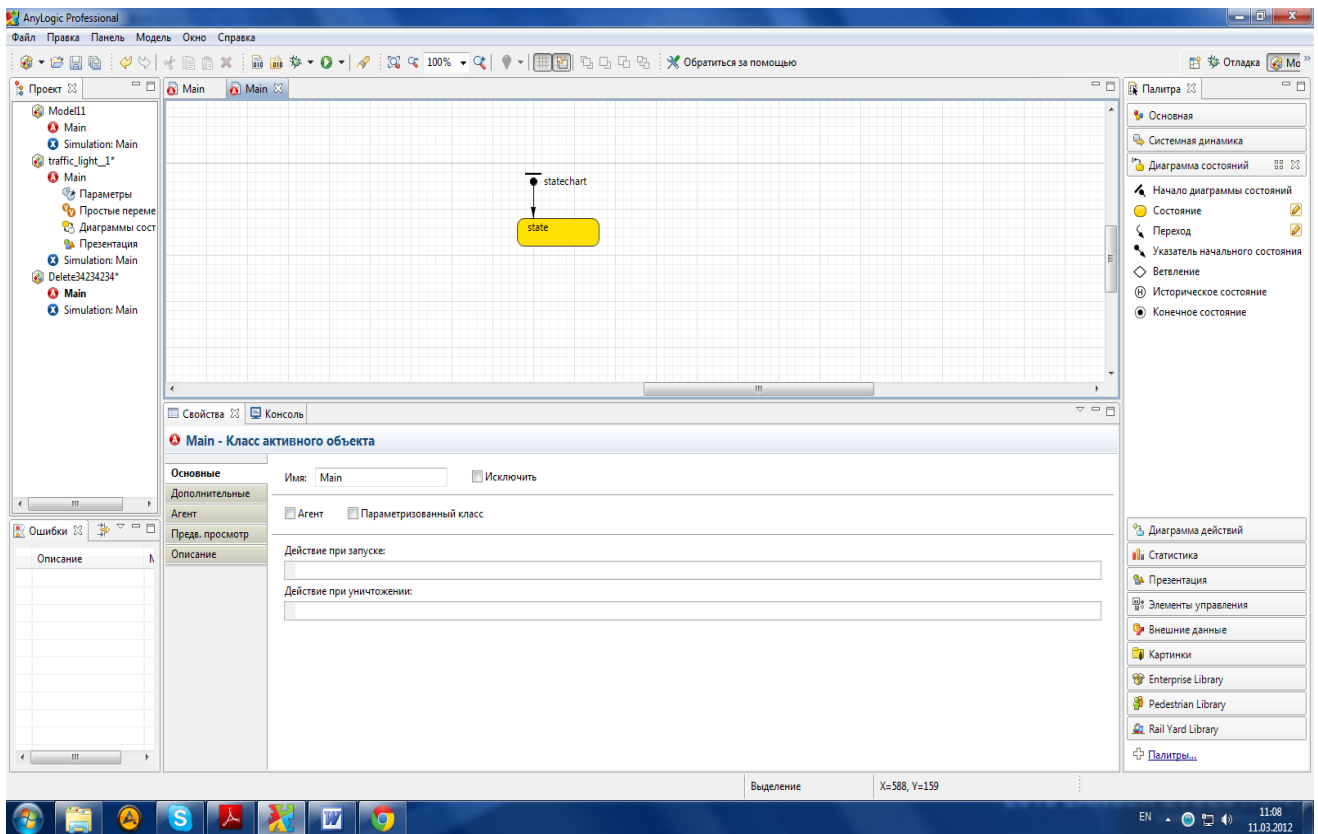


Рисунок 1.13 – Процес побудови діаграми станів

Ім'я стану, як і всі інші параметри, можна відредагувати в вікні його властивостей. Для того, щоб побудувати «стейтчарт», слід використовувати елементи з палітри **Діаграма станів**, рис.1.14.

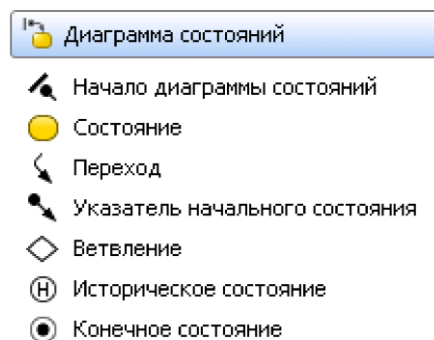


Рисунок 1.14 – Елементи палітри **Діаграми станів**

5. Тепер побудуйте наступну діаграму.

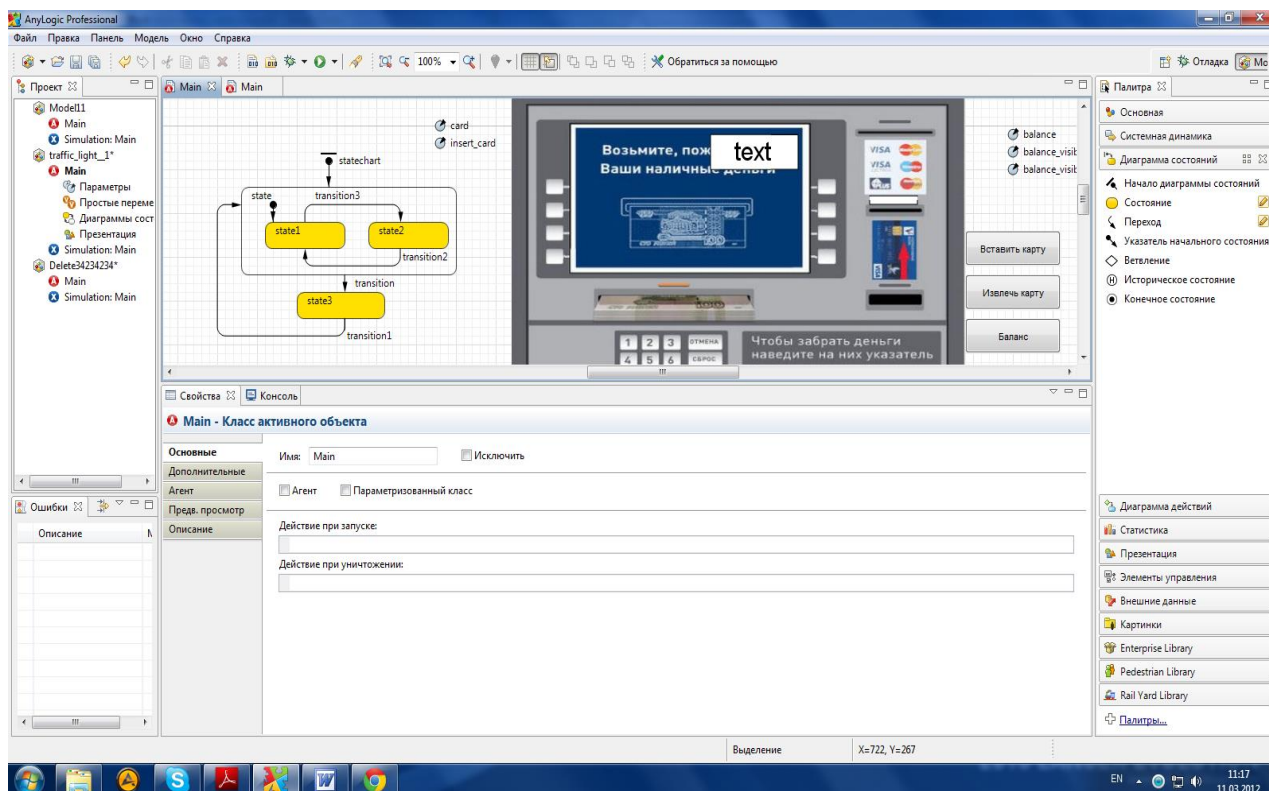


Рисунок 1.15 –Приклад готової діаграми станів

6. Для того, щоб задати умови спрацьовування переходів, виділяємо перехід transtion3, і в полі **Відбувається** залишаємо без зміни варіант **За таймаутом**, а в полі **По таймауту** вводимо 0.5 (рис. 1.15). Аналогічно виконуємо і з переходом transition 2.

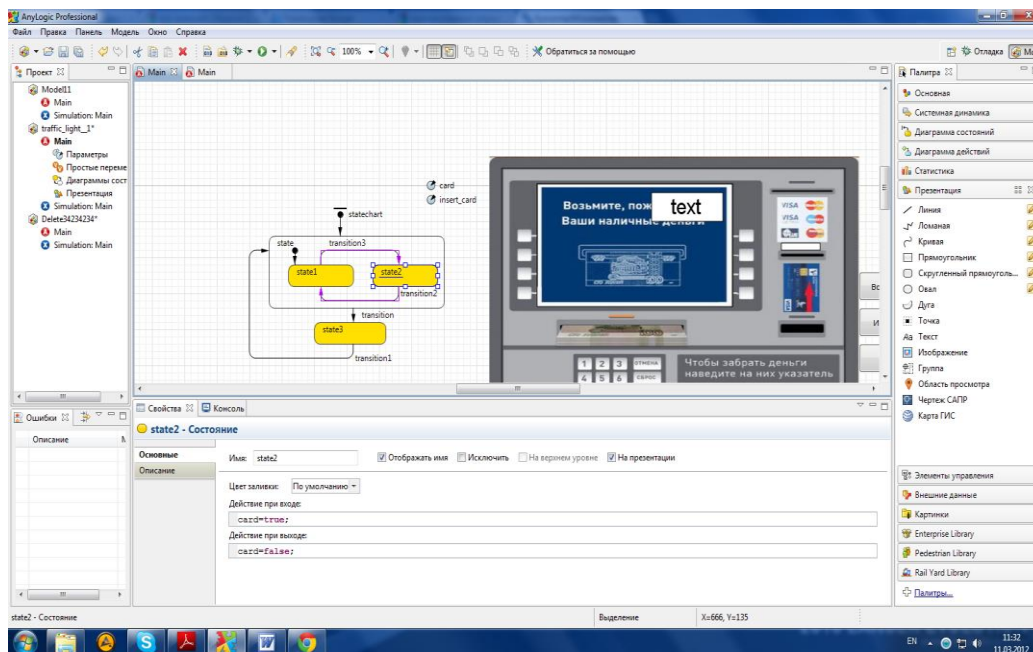


Рисунок 1.16 – Визначення параметрів State2

В перехід transition в полі **Відбувається** задаємо параметр **За виконання умови**, а в полі **За виконання умови** встановлюємо `insert_card == true`.

У transition1 задаємо аналогічні параметри transition, тільки в полі **За виконання умови** встановлюємо `insert_card == false`.

Об'єкту state2 задаємо параметри, як показано на рис.1.16.

7. Далі додамо на робочу область на банкомат об'єкт **Прямокутник** з вкладки **Презентація**. Саме цей прямокутник буде створювати ефект миготіння карто-приймача банкомату (рис. 1.17).

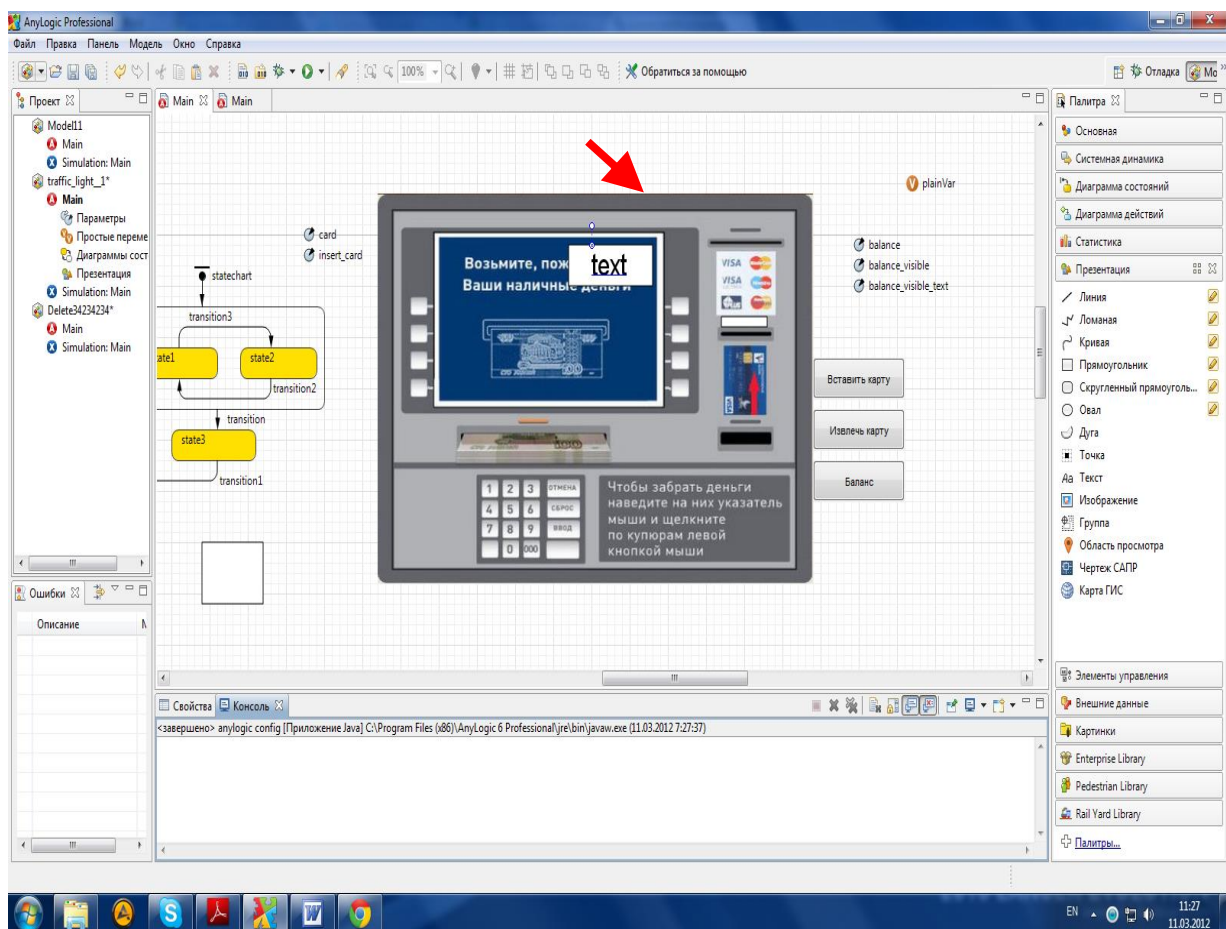


Рисунок 1.17 – Визначення основних елементів в рамках графічної реалізації моделі

У властивості **Динамічні-Колір заливки** даного прямокутника визначаємо наступне: `card? Color.green: Color.gray`.

8. На наступному кроці побудови даної моделі необхідно реалізувати імітацію зчитування банківської карти. Для цього додаємо три **Параметра** і визначаємо їх, як показано на рис. 1.18. У параметра balance задаємо **Тип** – `int`, а у `balance_visible` і `balance_visible_text` – `boolean`.

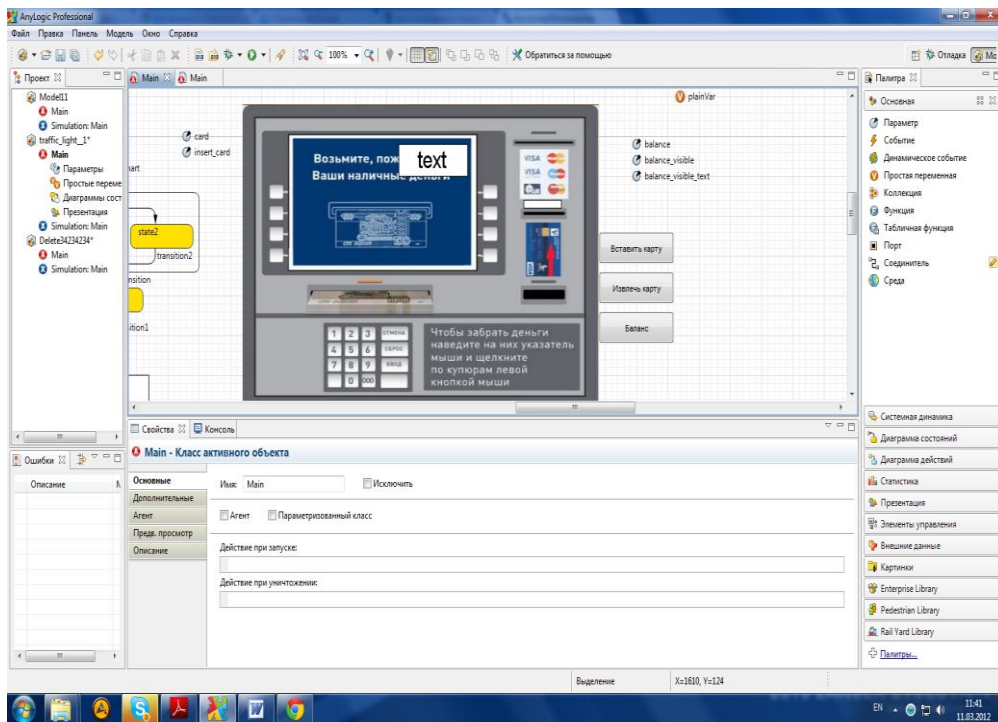


Рисунок 1.18 – Визначення параметрів моделі банкомату

9. Далі додаємо об'єкт **Текст** з вкладки **Презентація**, як показано на рис. 1.19.

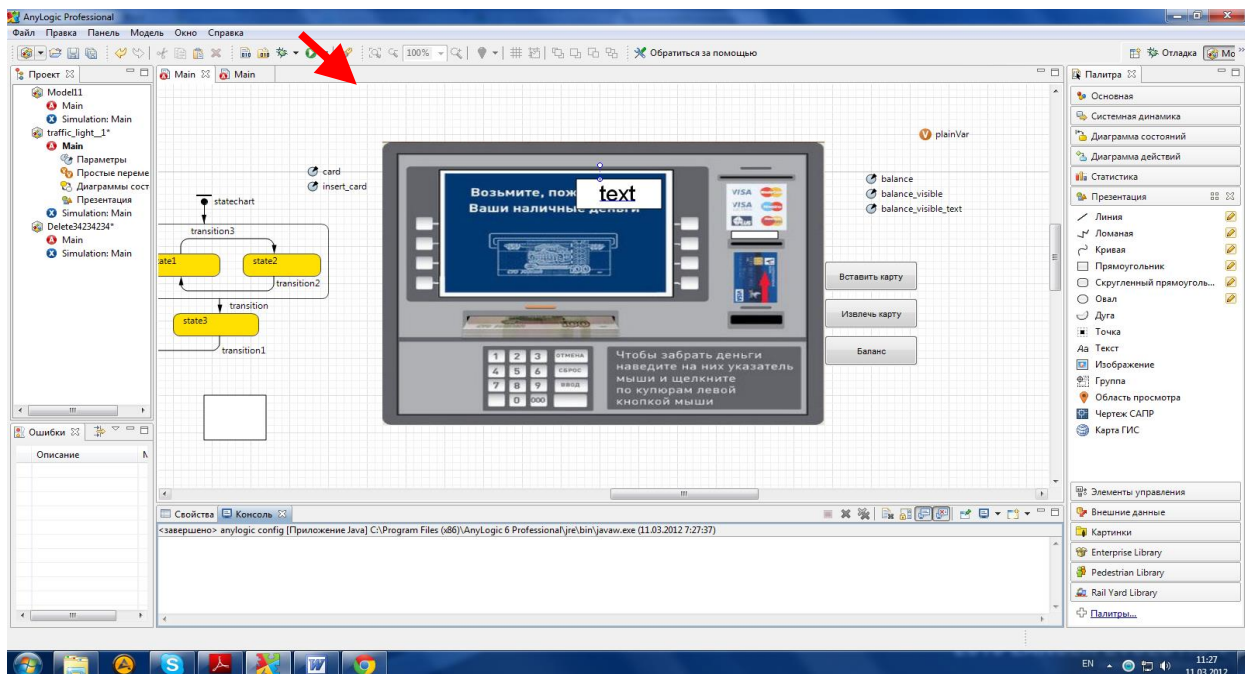


Рисунок 1.19 – Змінення графічного представлення моделі банкомату

10. Додаємо три **Кнопки** з **Палітри** – вкладки **Елементи управління**, для оперативного управління розглянутого банкомату (рис. 1.20).

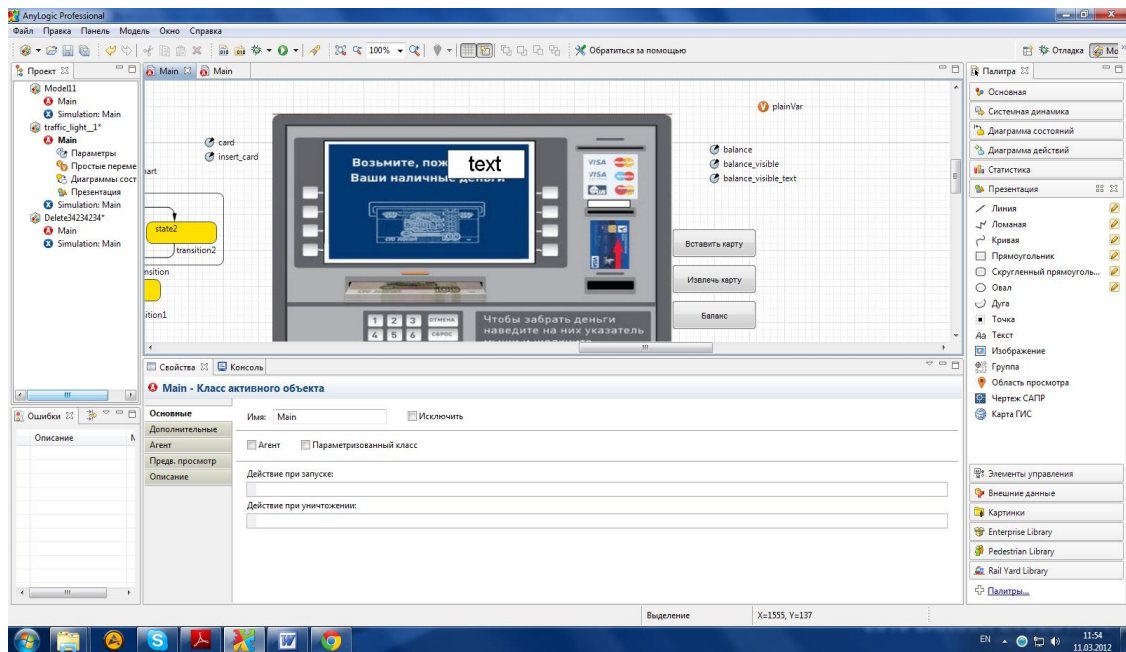


Рисунок 1.20 – Основні керуючі елементи моделі

Імена визначаємо за замовчуванням. У властивостях до дії кнопки **Вставте карту** додаємо код:

```
insert_card=true;
balance_visible=true;
balance=(int) (uniform(100, 10000));
button.setVisible(false);
```

Вийняти картку:

```
insert_card=false;
balance_visible=false;
balance_visible_text=false;
button.setVisible(true);
```

Баланс:

```
balance_visible_text=true;
```

При натисканні на кнопку **Вставити карту** кожного разу **Баланс** генерується випадковим чином. Функція `setVisible` встановлює видимість об'єкта.

Створена модель готова до виконання. Щоб її запустити натискаємо **Модель->Запустити** і далі ім'я вашої моделі.

Лабораторна робота №2

Тема роботи: конструювання об'єктів із застосуванням імітаційної системи моделювання.

Мета роботи: реалізувати в імітаційному пакеті **GPSS** модель розвитку епідемії.

Теоретичні відомості

GPSS (General Purpose Simulation System) – загальноцільова система моделювання складних систем, розроблена Джеффри Гордоном. Спочатку була розроблена і підтримувалася компанією IBM. На даний час є версії різних розробників. **GPSS World** – найсучасніша версія GPSS для персональних ЕОМ і ОС Windows. Розроблено компанією Minuteman Software.

Порядок виконання роботи.

Припустимо, що в мікрорайоні, населення якого становить 50000 чоловік, виникла епідемія, і з'явилося 80 інфекційних хворих. Припустимо, що приріст хворих за день пропорційний добутку числа здорових людей, які ще не переохворіли і не набули імунітет, на число хворих. Коефіцієнт пропорційності K_i (коефіцієнт поширення інфекції) включає різного роду профілактичні заходи, що вживають під час епідемії. Потрібно визначити, як швидко пошириться епідемія і яке буде максимальне число хворих.

1. Виявлення основних особливостей

Відомо, що процес розвитку епідемії може бути представлено системою двох диференціальних рівнянь:

$$X' = X \times (K_i \times Y - 1);$$

$$Y' = -K_i \times X \times Y,$$

де: X – хворі;

Y – здорові.

Зверніть увагу, що оскільки в записі рівнянь використовують однолітерні або дволітерні позначення, то треба застосовують позначення зі знаком підкреслення. Цей знак завжди слід вставляти для того, щоб введені позначення випадково не співпали зі стандартними числовими атрибутами, вбудованими в систему, і які несуть певну інформацію. Для моделювання епідемії необхідно виділити дві частини процесу: її моделюють як безперервний і як дискретний процес.

Результати моделювання представимо у вигляді графіка, в якому по осі абсцис відкладається час моделювання системи, а по осі ординат – число здорових і хворих людей в кожен момент часу моделювання системи.

Але перед цим необхідно обрати одиницю виміру часу. За таку одиницю приймемо день.

2. Створення імітаційної моделі процесу

Побудову імітаційної моделі почнемо зі створення заголовка моделі, який може бути представлено, наприклад, у такому вигляді:

; GPSSW – Yieaaiey.GPS
* Iiaaee.iaaiea yieaaiee *

Моделю будемо формувати з трьох секторів.

У першому секторі представимо частину процесу, яку моделюємо як безперервний процес. У цій частині введемо систему диференціальних рівнянь і початкові дані.

У другому секторі представимо частину процесу, яку моделюємо як дискретний процес.

У третьому секторі представимо праву частину системи звичайних диференціальних рівнянь за допомогою зовнішніх процедур вбудованої в систему GPSSW мови програмування PLUS.

Перший сектор моделі включає систему звичайних диференціальних рівнянь, що складається з двох рівнянь першого порядку. Окрім того, введемо початкові умови і необхідні вихідні дані. Систему диференціальних рівнянь можна представити в такому вигляді:

X INTEGRATE (Bolnoi())
Y INTEGRATE (Zdorov())

У дужках правої частини рівнянь визначено виклик процедур, які представляють собою праві частини відповідних рівнянь системи.

Опис цих процедур представлено в третьому секторі моделі.

Нижче зазначено початкові умови й інша інформація, яку вводять за допомогою оператора **EQU**. Параметр **Ki** представляє коефіцієнт поширення інфекції, який залежить від безлічі факторів.

У нашій задачі він дорівнює 0,00003.

Ki EQU 0.0001

Y EQU 50000

X EQU 80

Параметр Y_{-} – це число здорових людей на початку процесу моделювання.

Параметр X_{-} – це число хворих людей на початку процесу моделювання.

Другий сектор моделює часовий інтервал процесу моделювання.

Він дорівнює в нашій задачі 30 дням і може бути представлений в такому вигляді:

GENERATE 30

TERMINATE 1

Третій сектор моделі визначає праві частини диференціальних рівнянь системи за допомогою процедур вбудованої мови програмування для моделювання PLUS.

Оскільки в процесі інтегрування параметри, які нам потрібно відшукати, можуть вийти за допустимі межі, необхідно ввести відповідні обмеження за допомогою умовного оператора. Цю частину моделі можна представити так:

PROCEDURE Bolnoi () BEGIN ; Aeiaieea .inoa .enea aieuiuo.

TEMPORARY A;

IF (X < 0) THEN X = 0;

IF (X > 10e50) THEN X = 10e50;

A = X # (Ki # Y – 1);

RETURN A;

END;

PROCEDURE Zdorov() BEGIN ;

151

TEMPORARY B;

IF (Y < 0) THEN Y = 0;

IF (Y > 10e50) THEN Y = 10e50;

B = (0 – Ki_) # X # Y;

RETURN B_;

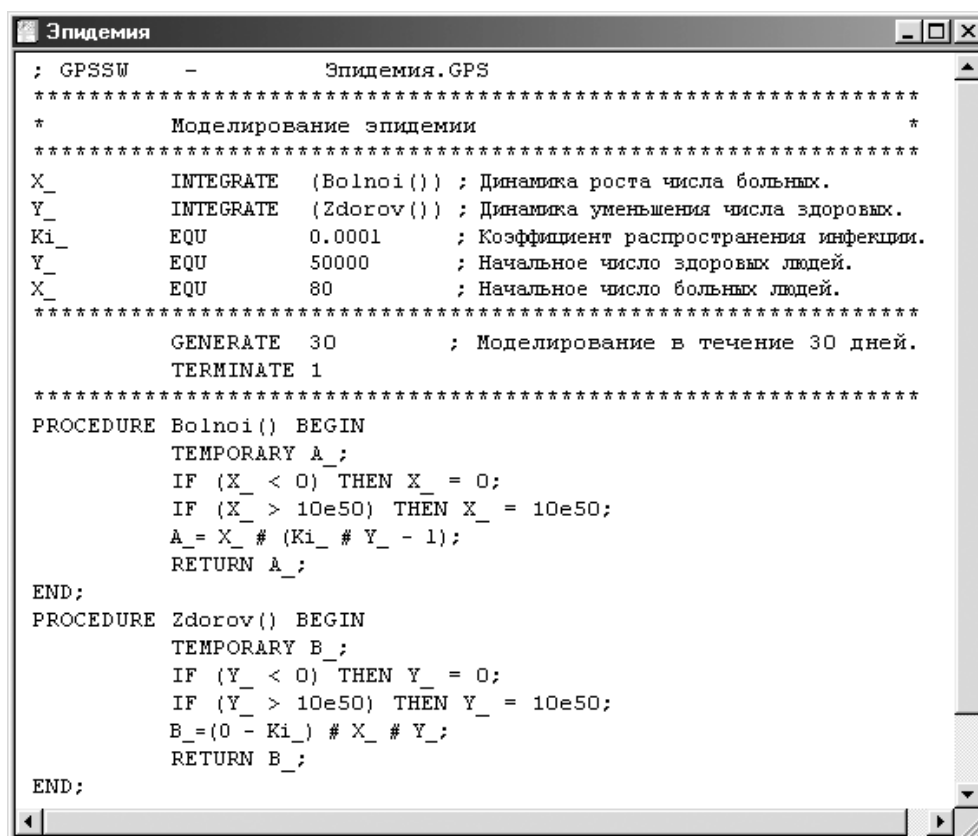
END;

3. Представлення імітаційної моделі

Для представлення імітаційної моделі виконаємо наступні дії:

- клацаємо пункт **File** головного меню системи. З'явиться меню, що випадає;
- клацаємо пункт **New** меню, що випадає. З'явиться діалогове вікно **Новий документ**;

- виділяємо пункт **Model** і натискаємо на кнопку **ОК**. З'явиться вікно моделі, в якому вводимо дану програму. Цей фрагмент моделі буде виглядати так, як показано на рис. 2.1.



```
; GPSSW - Эпидемия.GPS
*****
* Моделирование эпидемии *
*****
X_ INTEGRATE (Bolnoi()) ; Динамика роста числа больных.
Y_ INTEGRATE (Zdorov()) ; Динамика уменьшения числа здоровых.
Ki_ EQU 0.0001 ; Коэффициент распространения инфекции.
Y_ EQU 50000 ; Начальное число здоровых людей.
X_ EQU 80 ; Начальное число больных людей.
*****
GENERATE 30 ; Моделирование в течение 30 дней.
TERMINATE 1
*****
PROCEDURE Bolnoi() BEGIN
    TEMPORARY A_;
    IF (X_ < 0) THEN X_ = 0;
    IF (X_ > 10e50) THEN X_ = 10e50;
    A_ = X_ # (Ki_ # Y_ - 1);
    RETURN A_;
END;
PROCEDURE Zdorov() BEGIN
    TEMPORARY B_;
    IF (Y_ < 0) THEN Y_ = 0;
    IF (Y_ > 10e50) THEN Y_ = 10e50;
    B_ = (0 - Ki_) # X_ # Y_;
    RETURN B_;
END;
```

Рисунок 2.1 – Вікно імітаційної моделі поширення епідемії

Викликати вікно для подання імітаційної моделі в системі GPSSW можна також, натиснувши комбінацію клавіш **Ctrl + Alt + S**.

4. Моделювання системи

Після створення, імітаційну модель необхідно відтранслювати і запустити на виконання.

Для цього:

- клацаємо пункт **Command** головного меню системи або натискаємо комбінацію клавіш **Alt + C**. З'являється меню, що випадає;

- клацаємо пункт **Create Simulation** (Створити виконувану модель) випадаючого меню.

Далі переходимо до подання результатів рішення системи диференціальних рівнянь в графічному вигляді.

Для цього:

- клацаємо пункт **Window** головного меню системи або натискаємо комбінацію клавіш **Alt + W**. З'являється меню, що випадає;

- клацаємо пункт **Simulation Window** меню, що випадає. З'являється спливаюче меню;

- клацаємо пункт **Plot Window** спливаючого меню. З'являється діалогове вікно **Edit Plot Window**. Вводимо в це вікно інформацію, як показано на рис. 2.2.

Оскільки нам потрібно в графічному вікні розташувати два графіка, то заповнення діалогового вікна **Edit Plot Window** необхідно виконати в два етапи.

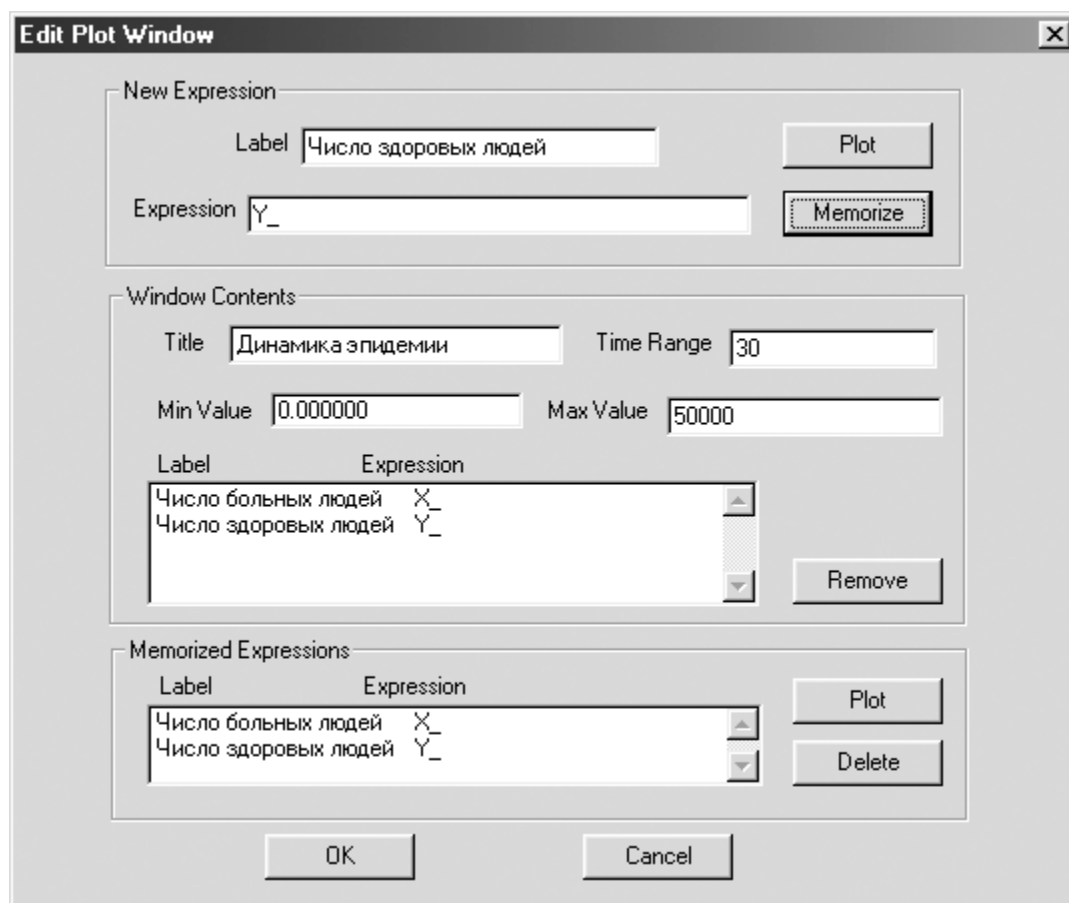


Рисунок 2.2 – Діалогове вікно Edit Plot Window для моделі поширення епідемії

На першому етапі введіть наступну інформацію в текстові поля, переміщаючись між ними за допомогою клавіші **Tab**:

- у полі **Label**, наприклад, фразу «Кількість хворих людей»;
- у полі **Expression** ім'я шуканої змінної $X_{_}$;
- у полі **Title**, наприклад, такий текст: «Динаміка епідемії»;
- у полі **Time Range** значення 30 – це число днів моделювання;
- у полі **Min Value** залишимо наведене значення без змін;
- у полі **Max Value** значення 50000 – це початкова кількість здорових людей.

Клацнемо кнопки **Plot i Memorize**. Інформація, яку було введено в перших двох текстових полях, з'явиться в двох багаторядкових списках, розташованих нижче.

Таким чином, ми вказали всю необхідну інформацію для виведення в графічному вигляді результатів моделювання, що стосуються шуканої змінної X – числа хворих людей.

Тепер ми надамо необхідну інформацію для виведення в графічному вигляді результатів моделювання, що стосуються шуканої змінної $Y_{_}$ – числа здорових людей.

Для цього введемо наступну інформацію в текстові поля, використовуючи клавішу **Tab** для переміщення між ними:

- в полі **Label** замість «Число хворих людей», наприклад, «Число здорових людей»;
- в полі **Expression** замість імені шуканої змінної $X_{_}$ ім'я іншої шуканої змінної – Y .

Далі:

- клацаємо кнопки **Plot i Memorize**. Дані, зазначені в перших двох текстових полях, додадуться до раніше введеної інформації в двох багаторядкових списках, розташованих нижче;
- клацнемо кнопку **OK**. З'явиться вікно **PLOTS** (рис. 2.3), із заготовкою (шаблоном) графіка.

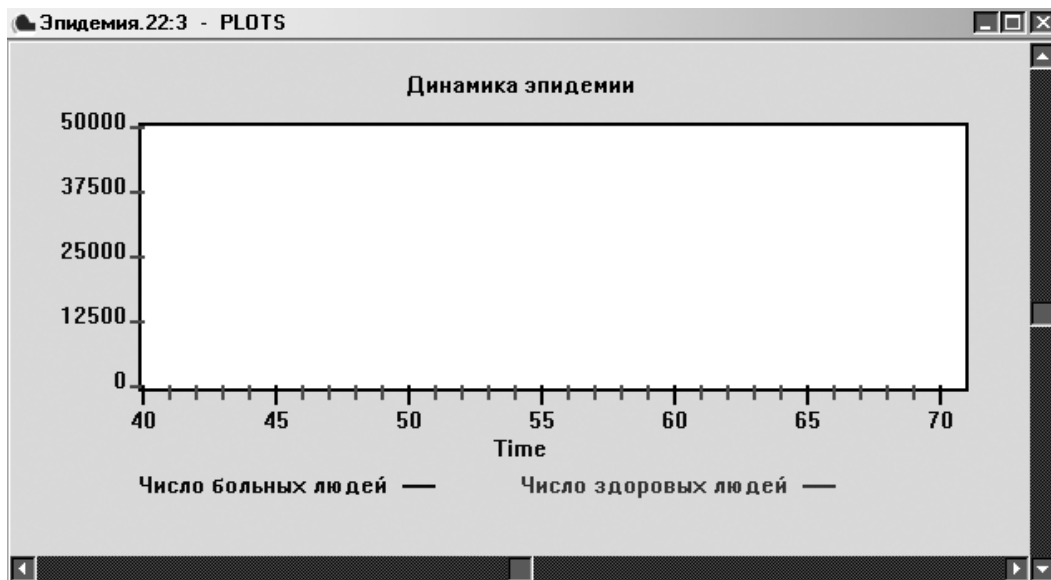


Рисунок 2.3 – Вікно PLOTS із заготовкою графіка для моделі поширення епідемії

Якщо заготовка графіка виглядає інакше, клацнемо горизонтальну смугу прокрутки графіка ліворуч від бігунка. З'явиться потрібна заготовка. Тепер можна запустити систему на моделювання.

Для цього:

- клацаємо пункт **Command** головного меню системи або натискаємо комбінацію клавіш **Alt + C**. З'являється меню, що випадає;
- клацаємо пункт **START** меню, що випадає. З'являється діалогове вікно **Start Command**;
- • клацаємо кнопку **OK**. Розпочинається рішення системи диференціальних рівнянь, наведеної в моделі.

Потім з'явиться вікно **PLOTS**, представлене на рис. 2.4, з шуканим графіком.

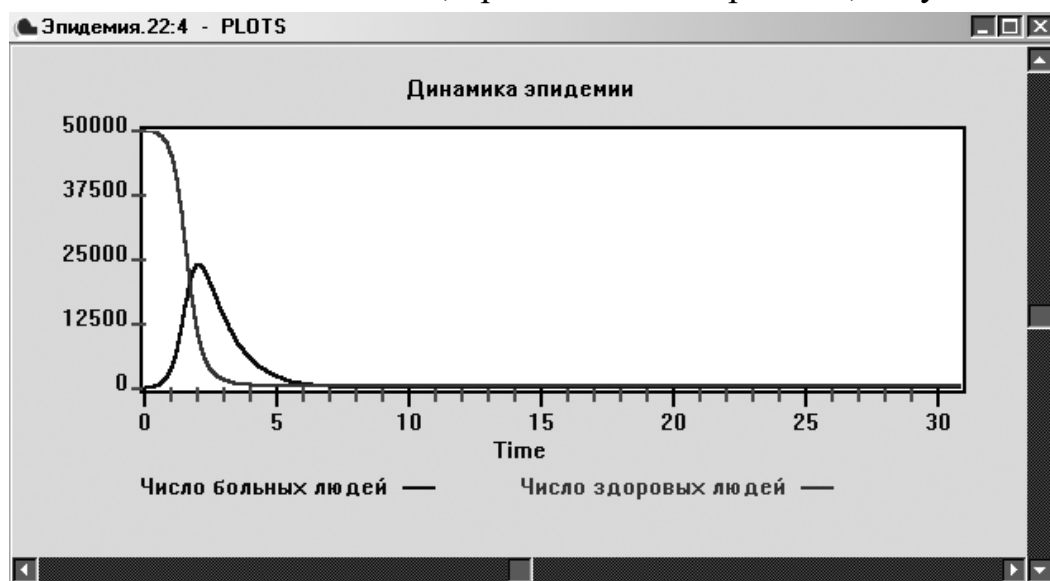


Рисунок 2.4 – Вікно PLOTS з графіком для моделі поширення епідемії

5. Аналіз отриманих результатів

Аналізуючи графічні дані, можна відзначити, що при заданих початкових значеннях – кількості здорових людей 50000, а хворих 801 – число хворих досягне максимуму через 2,5 дня і складе близько 25000 чоловік.

Давайте тепер подивимося, що буде, якщо, провівши профілактичні заходи, довести коефіцієнт поширення епідемії до значення 0,00003.

Щоб побачити результат цього рішення:

- вводимо нове значення коефіцієнта поширення епідемії в першому секторі моделі, замінивши значення 0,0001 на 0,00003. Це буде виглядати так:

Ki EQU 0.00003

- закриємо вікно **PLOTS**, натиснувши кнопку із зображенням хрестика в верхній правій частині вікна;

- клацнемо пункт **Command** головного меню системи або натиснемо комбінацію клавіш **Alt + C**. З'явиться меню, що випадає;

- клацнемо пункт **Retranslate** (Перетранслювати) меню, що випадає;

- клацнемо пункт **Window** головного меню системи або натиснемо комбінацію клавіш **Alt + W**. З'явиться меню, що випадає;

- клацнемо пункт **Simulation Window** меню, що випадає. З'явиться спливаюче меню;

- клацнемо пункт **Plot Window** спливаючого меню. З'явиться діалогове вікно **Edit Plot Window**. Тепер можна встановити параметри для графіка, які були збережені минулого разу за допомогою кнопки **Memorize**;

- виберемо в діалоговому вікні **Edit Plot Window** в розділі **Memorized Expressions** рядок з шуканою змінною: «Число хворих людей X» і натискаємо кнопку **Plot**;

- виберемо в цьому ж розділі інший рядок з шуканою змінною «Число здорових людей Y» і натискаємо кнопку **Plot**;

- вводимо у текстовому полі **Time Range** (Тимчасова область) колишнє значення 30;

- вводимо у текстовому полі **Max Value** (Максимальна величина) колишнє значення, яке дорівнює 50000. Для переходу до іншого текстового поля можна використовувати мишу;

- клацнемо пункт **Command** головного меню системи або натиснемо комбінацію клавіш **Alt + C**. З'явиться меню, що випадає;

- клацнемо пункт **START** меню, що випадає. З'явиться діалогове вікно **Start Command**;
- клацнемо кнопку **OK**. З'явиться вікно **PLOTS** рис. 2.5, з шуканим графіком.

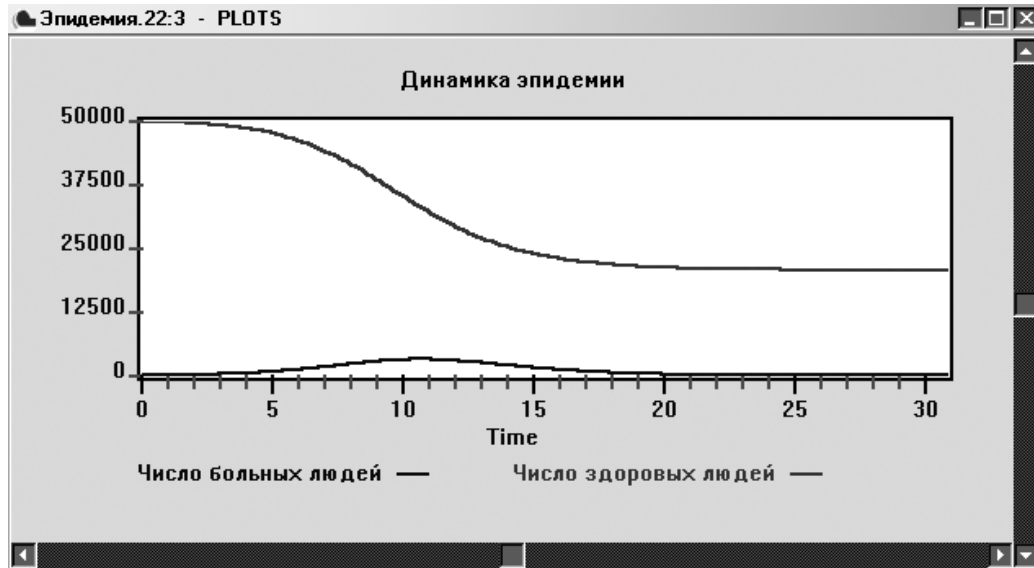


Рисунок 2.5 – Графік для моделі поширення епідемії після зміни початкових параметрів

Таким чином, маючи задані початкові значення здорових і хворих людей, відповідно 50000 і 80, число хворих досягне максимуму на 11 день і становитиме 4000. Це в шість разів менше, ніж у першому випадку.

Можете поекспериментувати і з іншими значеннями коефіцієнта поширення епідемії.

Лабораторна робота №3

Тема: конструювання об'єктів із застосуванням імітаційної системи моделювання.

Мета: вивчити можливості використання **Arena** для моделювання явищ в децентралізованих системах (на прикладі моделювання роботи системи обслуговування покупців на касі супермаркету).

Теоретичні відомості

Arena – система імітаційного моделювання, яка дозволяє створювати динамічні моделі різнорідних процесів і систем, оптимізувати побудовану модель. Програма Arena забезпечена зручним об'єктно-орієнтованим інтерфейсом, має широкі функціональні можливості по адаптації до різних предметних областей.

Основою технології моделювання Arena є мова моделювання SIMAN та анімаційна система Cinema Animation. Відрізняється гнучкими і виразними засобами моделювання. Процес моделювання має таку структуру, а саме:

Спочатку користувач крок за кроком будує в візуальному редакторі програми Arena **модель**. Потім система генерує її відповідний код на SIMAN, після чого відбувається автоматичний запуск Cinema Animation.

Arena складається з блоків моделювання (модулів) та операцій (сутностей). Сутності рухаються між модулями в міру їх обслуговування.

Порядок виконання роботи:

Змоделюємо роботу системи обслуговування покупців на касі супермаркету, якщо відомо, що потік покупців має пуассонівський розподіл із середнім значенням 5 хвилин (позначається POIS (5)), а час обслуговування на касі займає від 2 до 10 хвилин з найбільш імовірним значенням 3 хвилини (використовується розподіл Triangular). Який середній час очікування покупців у черзі, якщо тривалість моделювання становить 15 годин?

1. Перемістимо модулі Create, Process і Dispose до вікна робочого модуля (рис.3.1, 3.2, 3.3).

Для завдання властивостей кожного графічного модуля двічі клацнемо по ньому, і в діалозі задамо значення параметрів відповідно до умови.

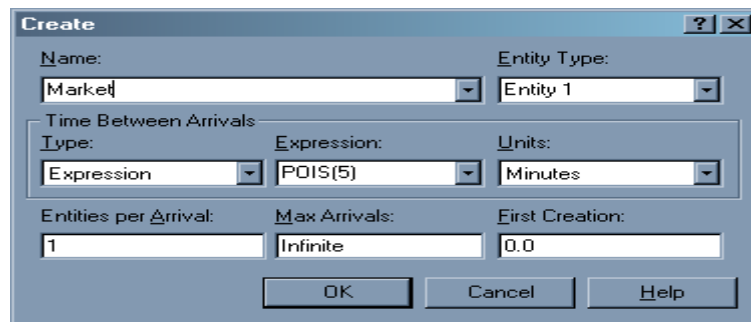


Рисунок 3.1 – Діалогове вікно властивостей модуля Create

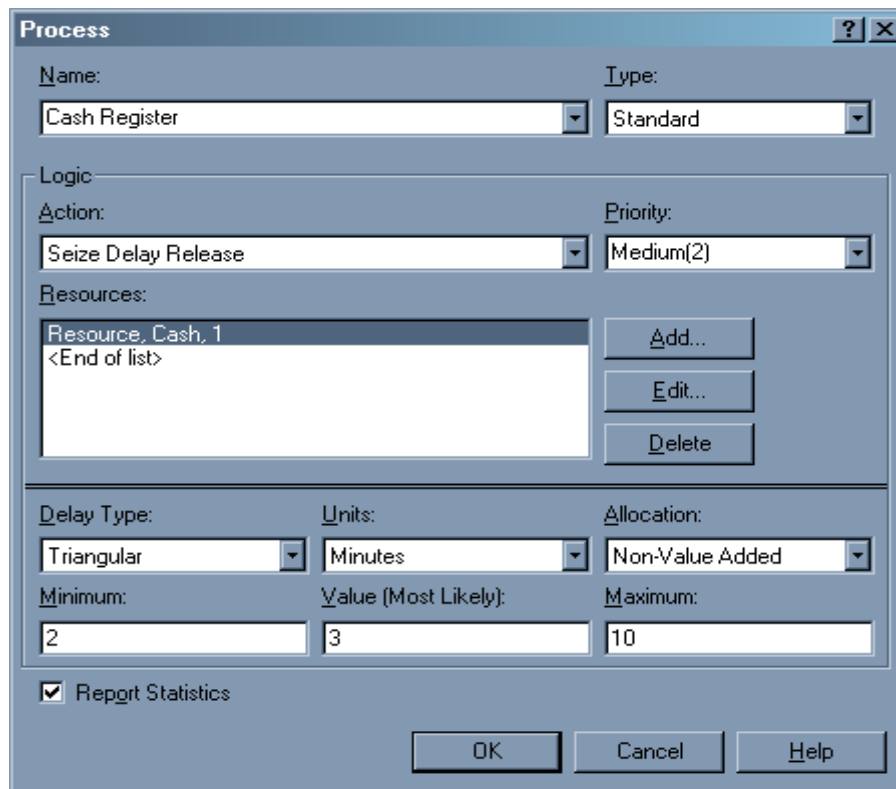


Рисунок 3.2 – Діалогове вікно властивостей модуля Process

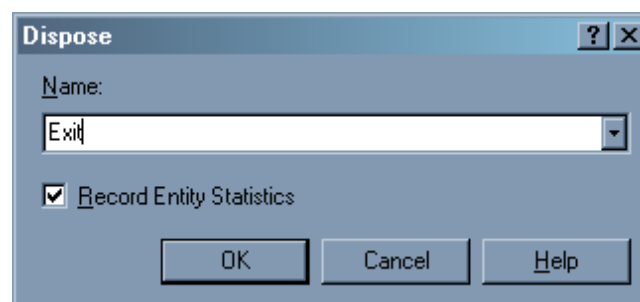


Рисунок 3.3 – Діалогове вікно властивостей модуля Dispose

Після завантаження кожного модуля модель приймає вигляд (рис. 3.4):

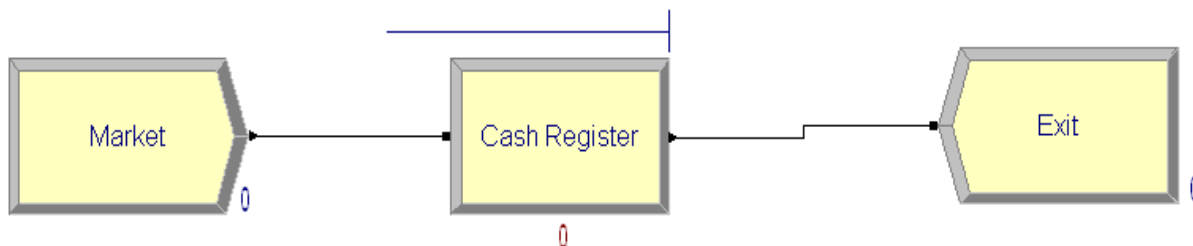


Рисунок 3.4 – Імітаційна модель системи обслуговування покупців на касі супермаркету

2. Для завантаження тривалості моделювання перейдемо в меню Run / Setup. У полі Replication Length встановимо тривалість 900, а в полі Time Units одиницю виміру часу Minutes. В Base Time Units також вказуємо Minutes для генерації звіту в хвилинах (рис. 3.5).

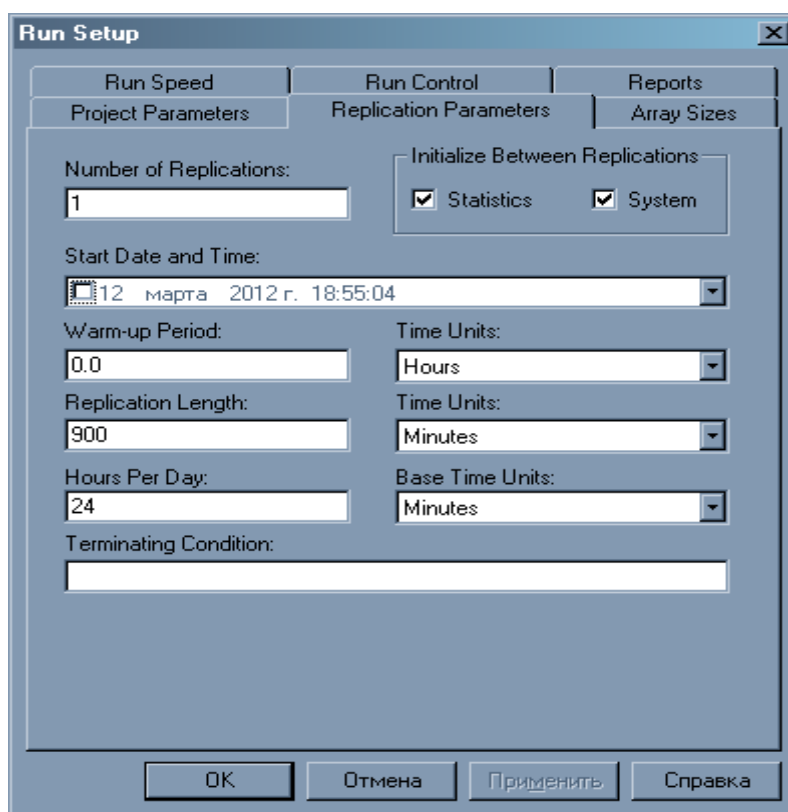


Рисунок 3.5 – Вікно параметрів моделювання

Таблиця 3.1. Результати моделювання моделі

Характеристика	Де знайти	Значення
Середня тривалість очікування покупців в черзі	Queue – Time – Waiting Time (Average)	16,9
	<u>Waiting Time</u>	хвилин
	Cash Register.Queue	16.9295

Лабораторна робота №4

Тема: моделювання об'єктів за допомогою інформаційних систем імітаційного моделювання.

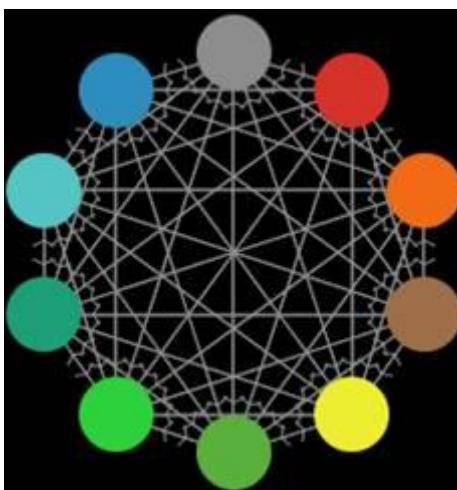
Мета: вивчити можливості використання **NetLogo** для моделювання явищ в децентралізованих системах.

Теоретичні відомості

Мова NetLogo був створений Урі Віленський в 1999 році і продовжує активно розвиватися в даний час. Середовище програмування NetLogo служить для моделювання ситуацій і феноменів, що відбуваються в природі і суспільстві. NetLogo зручно використовувати для моделювання складних, що розвиваються у часі систем. ворець моделі може давати вказівки сотням і тисячам незалежних «агентів» що діють паралельно. Це відкриває можливість для пояснення і розуміння зв'язків між поведінкою окремих індивідуумів і явищами, які відбуваються на макро рівні.

Завдяки потужним обчислювальним засобам і відносній простоті синтаксису NetLogo, на його основі в останні роки було побудовано безліч дослідних моделей, які використовувалися і обговорювалися в книгах по багатоагентного моделювання та моделювання в соціології.

Важливою особливістю четвертої версії мови NetLogo є поява нового типу агентів. До черепашек (turtles) і цяток (patches) додалися зв'язки (links). Агенти нового типу відкривають нові можливості для моделювання мережних відносин. Зв'язок в NetLogo це – агент, що зв'язує дві черепашки або два вузла.



Створення спрямованих зв'язків в середовищі NetLogo

Порядок виконання роботи:

Ознайомлення з прикладом – моделлю «Синхронізація світлячків» (моделі / Fireflies.nlogo).

Аналіз роботи моделі формування колонії світлячків (моделі / Slime.nlogo). Виконати завдання, зазначені в пункті «Те, що варто спробувати» вкладки Information. Удосконалити модель відповідно до пункту «Розвиток моделі». Запропонуйте власний варіант розвитку. Як його можна реалізувати? Зберегти нову модель і скріншоти, що ілюструють поведінку системи.

Робота з індивідуальною моделлю (за тим самим планом).

Складання звіту про роботу, що має необхідні пояснення та фото екрану.

Приклад роботи з моделлю «Синхронізація світлячків».

Світлячки (сімейство жучків Lampyridae, понад 2000 видів) відомі своєю здатністю випускати в темряві м'яке світло, що фосфоресціює. Коефіцієнт корисної дії «ліхтариків» світлячків надзвичайно високий. Якщо у звичайній електричній лампочці в видиме світло перетворюється близько 5% енергії (а решта розсіюється у вигляді тепла), то у світлячків в світлові промені переходить від 87 до 98% енергії. Процес світіння знаходиться під нервовим контролем.

Багато видів здатні за своїм бажанням зменшувати і збільшувати силу світла, а також випускати переривчасте світло.

Вважають, що біоломінесценція світлячків – засіб міжполового спілкування: світловими сигналами партнери повідомляють один одному про своє місцезнаходження. Деякі види світлячків здатні спалахувати і затухати в унісон всією зграєю, що зібралася на одному дереві. Жучки, що розташувалися на сусідньому дереві, теж спалахують узгоджено, але не в такт зі світлячками, які сидять на першому дереві. Також, у своєму власному ритмі, світяться жучки й на інших деревах.

Година за годиною, тижнями і навіть місяцями, жучки блимають на своїх деревах в одному ритмі. Ні вітер, ні дощ не можуть змінити інтенсивності і частоти спалахів. Можливо порушити синхронність спалахів тільки у тому випадку, якщо висвітлити дерево яскравою лампою. Але, коли зовнішній подразник згасне, світлячки знову, немов по команді, почнуть блимати. Спочатку ті, що в центрі дерева, пристосовуються до одного ритму, потім сусідні жуки підключаються до них, і поступово хвилі, спалахуючих в унісон вогників, поширюються по всіх гілках дерева.

Розглянемо групову поведінку світлячків (синхронізацію їх мерехтінь) і спробуємо змодельовати його за допомогою NetLogo. NetLogo – це вільне програмне забезпечення для агентного моделювання, що використовує JVM і працює на більшості платформ.

1. Початок роботи: Command Center і створення проекту

Почнемо з роботи з Command Center – вікном, розташованим в нижній частині вкладки Interface. Подібно Command Window в MATLAB, він призначений для введення команд і перегляду результатів їх виконання. Створимо 100 черепашок (так називають агентів, здатних переміщатися ігровим світом) (рис.4.1).

```
observer> crt 100
```

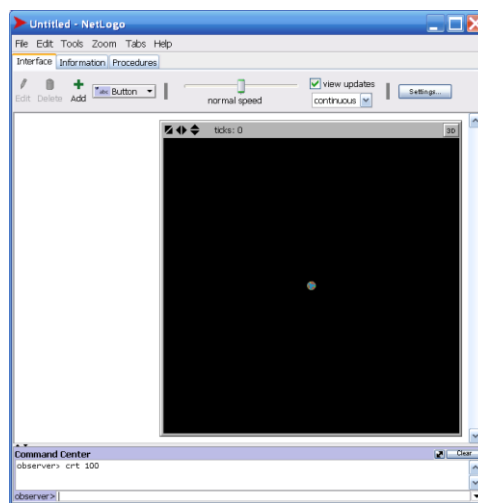


Рисунок 4.1

Всі створені черепашки мають одні й ті самі координати і розташовані в центрі світу (можна уявити собі «стопочку» черепашок, що сидять одна на іншій). Кожна черепашка зорієнтована в своєму напрямку (за замовчуванням черепашок зображують стрілкою, напрямком якої вказує на орієнтацію черепашки). Новостворені черепашки орієнтовані випадковим чином. Колір черепашок також вибирають випадково.

Щоб показати, що черепашок дійсно сотня, потрібно зрушити їх від центру (рис. 4.2). Оскільки ця команда стосується тільки черепашок (turtles), змінимо звернення `observer>` на `turtles>`, і введемо наступне

```
turtles> fd 10
```

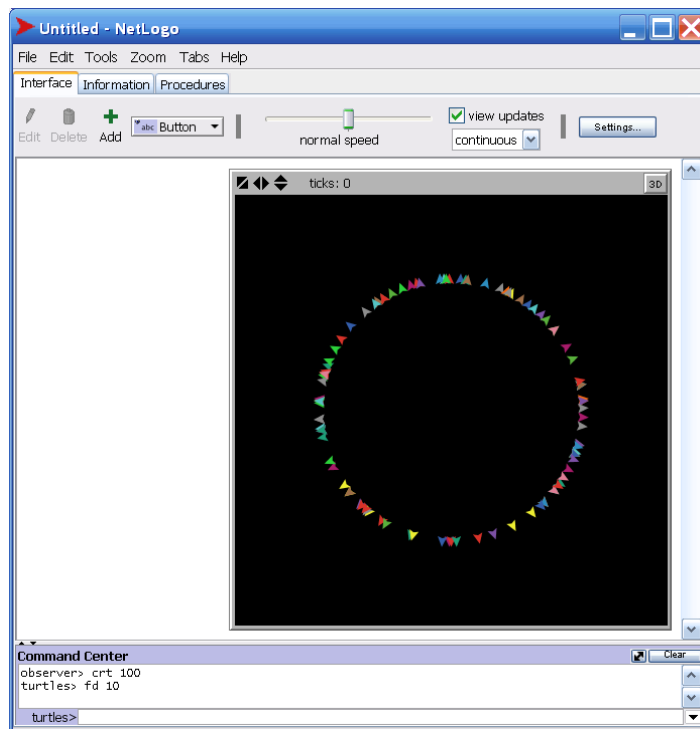


Рисунок 4.2

Черепашки перемістилися вперед на 10 кроків. Тепер накажемо їм переміститися вперед на випадкове число кроків (від 0 до 9) (рис.4.3):

turtles> fd (random 10)

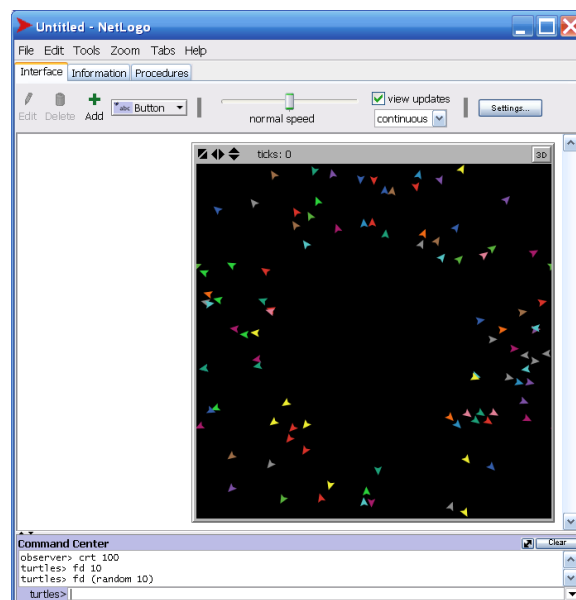


Рисунок 4.3

Нарешті, перейшовши знову до observer>, очистимо ігровий світ командою **ca** – всі черепашки зникнуть.

Набирати поодинокі команди в **Command Center** зручно, але навряд чи цей спосіб підійде для створення скільки-небудь складної моделі (особливо, якщо ми захочемо її потім використовувати). **NetLogo** дозволяє створювати власні програми та процедури. Почнемо зі створення кнопки **setup**, призначення якої – поставити початкові установки моделі (рис.4.4).

Для цього klikнемо іконку **Button** на панелі інструментів, а потім – на вільному місці у вкладці **Interface**, там, де ми хочемо створити кнопку. У віконці, що з'явилося введемо **setup** у полі **Commands** (команди, які будуть виконуватися при натисканні кнопки) і в полі **Display Name** (напис, що відображається на кнопці). Тиснемо **OK**.

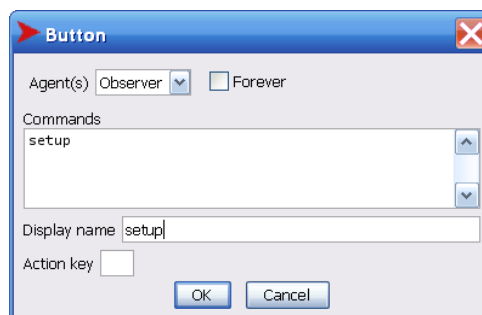


Рисунок 4.4

Однак ім'я кнопки і ім'я вкладки **Interface**, на якій вона розташована, підсвічено червоним. Це означає, що десь є помилка (рис. 4.5).

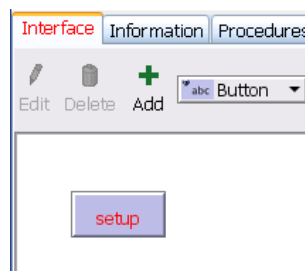


Рисунок 4.5

Помилку вказано тому, що ще не створена процедура, яка буде виконуватися після натискання кнопки (командами або примітивами частіше називають вбудовані команди NetLogo, а створені користувачем команди називають процедурами. Одні й інші мають однаковий статус).

Отже, перейдемо до вкладки **Procedures**, щоб створити процедуру **setup** (рис. 4.6).

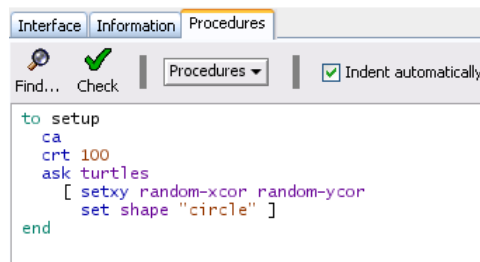


Рисунок 4.6

Розберемо цей код докладніше:

ca (скорочення від **clear-all**: можна використовувати як повне, так і скорочене ім'я команди). Ця команда, як ми вже знаємо, очищає ігровий світ.

crt 100 – знову відома команда – створює 100 черепашок.

ask turtles [] вказує (буквально: просить) кожну черепашку незалежно від інших виконувати команди, розташовані в дужках. Це еквівалентно тому, як ми вибирали **turtles>** в командному рядку. Відзначимо, що **crt** не розташований в дужках, а значить не має відношення до черепашок. Ця команда належить до іншого виду агентів – спостерігачів (**observer**). Якщо в самій команді або в блоці **ask** не вказано вид агентів, до яких відноситься команда, значить її виконує спостерігач. У NetLogo існує єдиний спостерігач, якого можна уявити собі як якусь «верховну істоту», що розглядає ігровий світ зовні, і яка бачить усе навкруги, що відбувається. Третій вид агентів – ділянки (**patches**) – нерухомі прямокутні ... ділянки, які служать для створення ландшафту ігрового світу. Четвертий вид агентів – зв'язки, що з'єднують між собою двох і більше черепашок. Останні два види агентів ми тут розглядати не будемо.

set shape «circle» задає форму (**shape**), тобто графічне представлення черепашок. Дана команда змінює форму черепашок з прийнятої за замовчуванням на кругову (**circle**). Створювати нові графічні уявлення можна за допомогою **Shapes Editor** (в меню **Tools**. Свої редактори є для черепашок і зв'язків).

Нарешті, **setxy random-xcor random-ycor** насправді являє собою відразу кілька команд. Спочатку черепашки виконують команди **random-xcor** і **random-ycor**, кожна з яких повертає випадкове число в діапазоні від мінімального до максимального значення координат по осі **x** і **y** відповідно. Потім ці два числа використовують як аргументи команди **setxy**.

Натиснемо кнопку `setup` і отримаємо очікувану картинку (рис. 4.7).

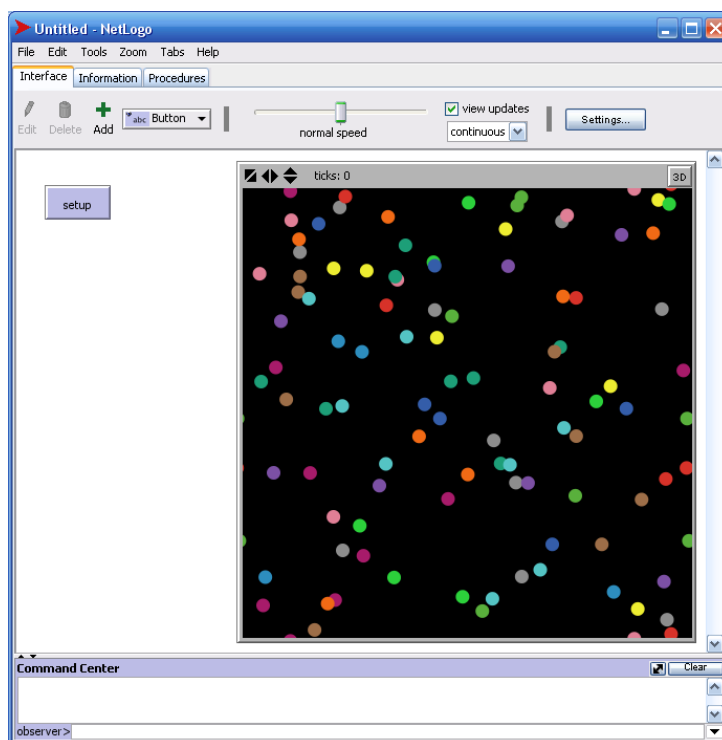


Рисунок 4.7

Кожного разу, натискаючи `setup`, ми будемо отримувати новий розподіл черепашок.

Перед тим, як перейти до створення наступної процедури, має сенс зберегти проект (File / Save).

Отже, приступимо до створення кнопки `go-one-step` (рис.4.8). Нехай процедура, яку викликають натисканням кнопки, називається `go` (взагалі кажучи, між написом на кнопці і ім'ям процедури немає ніякого зв'язку).

Перемикаєм у вкладку **Procedures** і запишемо код процедури

```
to go  
  move  
end
```

Але що таке `move`? Це що, примітив як `fd`? Ні, це наступна процедура, яку нам належить написати:

```
to move  
  ask turtles  
  [set heading random 360  
   fd 1]  
end
```

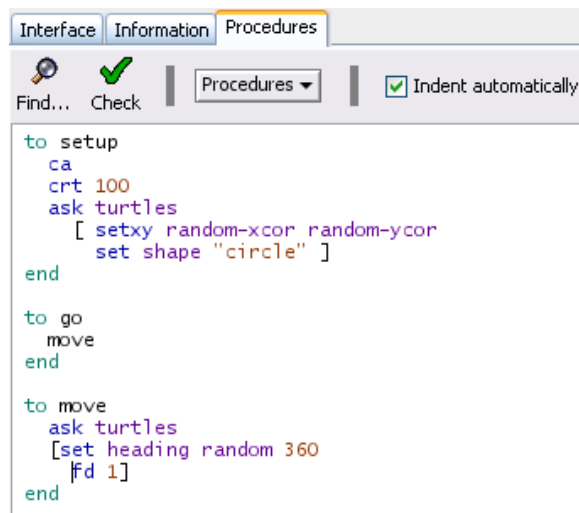



Рисунок 4.8

Тепер розглянемо код цієї процедури:

`ask turtles []` – визначає, що робити черепашкам.

`set heading random 360` – ще одна «двокрокова» команда. Перш за все, кожна черепашка вибирає випадкове ціле число в діапазоні від 0 до 359 (`random` не містить числа, яке є його аргументом). На другому етапі це число присвоюють орієнтації черепашки (`heading`, якщо хочете – напрямку «голови» ючерепашки). Орієнтацію вимірюють в градусах, які відраховують за годинниковою стрілкою.

`fd 1`: Тепер черепашки повинні зробити один крок вперед у новому напрямку.

Хіба не можна написати це прямо всередині **go**? – звичайно можна, але наш проект буде розвиватися, і в нього будуть додаватися нові можливості. У той же час хотілося б зберегти процедуру **go** простою, наскільки це можливо, щоб потім зручно можна було розібратися, як влаштована модель. Поступово `go` будуть додаватися розрахункові процедури, виведення графіків та інше. І кожен таку дію буде реалізовано окремою процедурою.

Кожен раз, коли ми натискаємо **go-one-step**, **turn-step** буде виконуватися. Було б добре, щоб виконання **turn-step** тривало стільки кроків, скільки нам потрібно. Домогтися цього легко: створюємо нову кнопку – **go** (вона виконує процедуру **go**), і у вікні редагування відзначаємо галочкою **Forever** (рис.4.9).

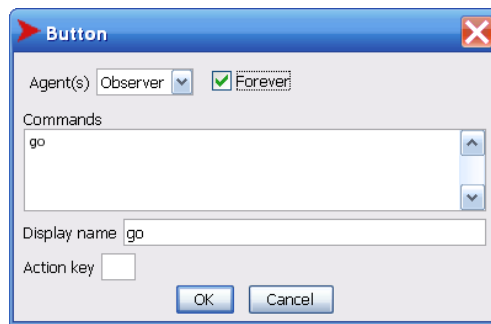


Рисунок 4.9

В результаті буде створена «постійна кнопка» (forever-button) **go**, яка буде залишатися включеною до того часу, поки її не вимкнуть повторним натисканням.

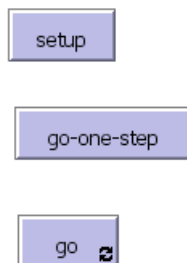


Рисунок 4.10

2 Робота з командами

Давайте поки поекспериментуємо з іншими командами, що стосуються черепашок. Наприклад, наберіть в **Command Center** **set color red** (не забули вказати адресата – turtles>?), або додайте це в код процедур (в блок ask turtles [commands]). Зауважте, що turtles> set color red еквівалентно, наприклад, наступному: observer> ask turtles [set color red].

Ви може набрати turtles> pendown (опустити перо, щоб черепашка могла креслити траєкторію свого руху) і побачите «сліди», що залишаються черепашками, або змінити set heading (random 360) на lt (random 45). Можна зробити число створюваних черепашок змінним. Замість crt 100 в setup запишемо crt number, а значення змінної number задамо за допомогою повзунка, який можна створити так само, як і кнопку. Добре те, що дуже швидко можна побачити результати змін. Проте, робота над проектом триває

3 Розробка моделі: від черепашок до світлячків

У нас є сотня черепашок безцільно блукаючих ігровим світом. Хотілося б, щоб черепашки зображували світлячків і наслідували, нехай і спрощено, їх поведінку. Наприклад, мерехтіли синхронно при зустрічі один з одним. Для

цього потрібно забезпечити наших черепашок-світлячків декількома змінними: clock – «внутрішній годинник» світлячка (він буде світити при clock = 0, потім показання лічильника поступово збільшуються до 9, після чого знову наближуються до 0), threshold – значення clock при якому світлячок перестає світити, reset-level – показання лічильника, які виникають, коли світлячок бачить іншого світлячка який світиться, і window – період «байдужості» після спалаху, протягом якого світлячок не реагує на спалахи інших.

Почнемо з того, що оголосимо ці змінні перед кодом процедур.

```
turtles-own
```

```
[Clock;; лічильник світлячка threshold;; значення лічильника при якому  
світлячок перестає світити
```

```
reset-level;; показання лічильника, що виникають, коли світлячок бачить  
спалах
```

```
window;; світлячок не реагує на спалахи, якщо (clock <= window)
```

```
]
```

Після «;» йде коментар до коду.

Тепер ми можемо записати процедуру setup, де дамо оголошеним змінним реалістичні значення. Світлячки, які світять, зображуються у жовтому кольорі, а решта – у сірому.

```
to setup
```

```
  ca
```

```
  crt number
```

```
  ask turtles
```

```
    [ setxy random-xcor random-ycor
```

```
      set shape «circle»
```

```
      set clock random (round 10) ;; тому що значення лічильника змінюються  
від 0 до 9
```

```
      set threshold 1
```

```
      set reset-level threshold
```

```
      set window -1
```

```
      ifelse (clock < threshold)
```

```
        [ set color yellow ]
```

```
        [ set color gray ]
```

```
      ]
```

```
end
```

Ми використовували нову команду: `ifelse (test) [commands1] [commands2]`, яка виконує команди `commands1`, якщо значення `test` істинно, і `commands2` в іншому випадку.

Вводимо все це (рис. 4.11).

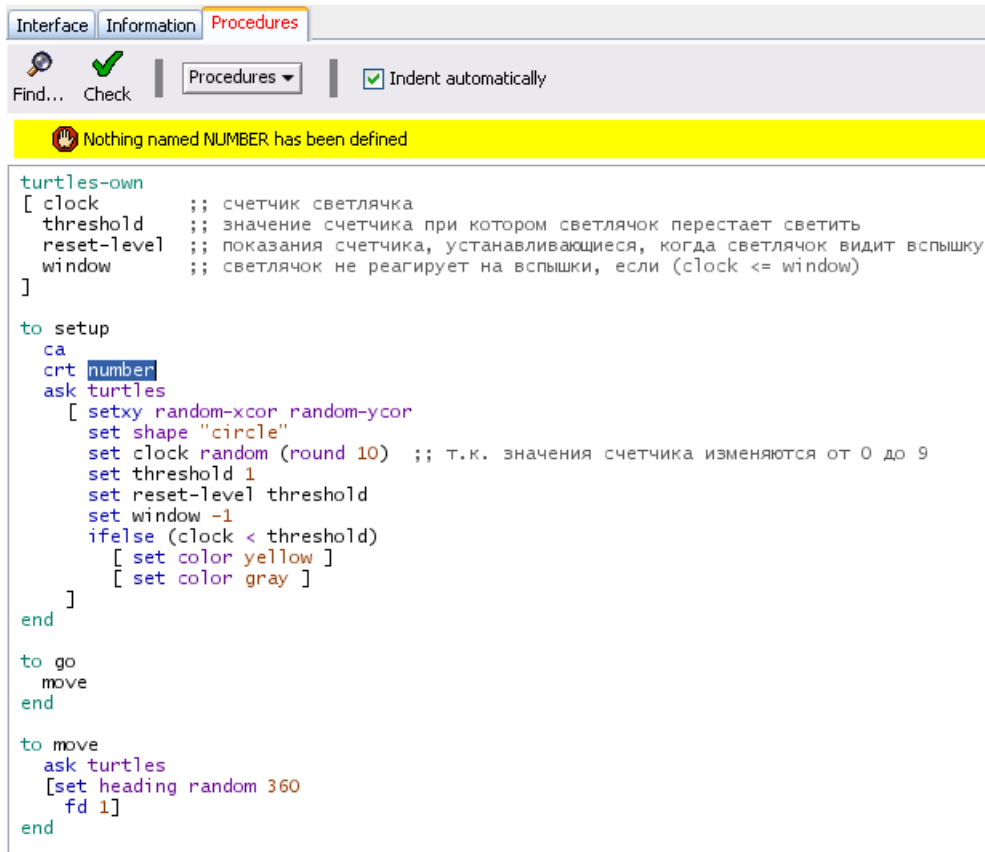


Рисунок 4.11

Що ж, знову лайка – значення-то `number` ми ще не задали! Виправляємо. Використовуємо кнопку на панелі інструментів вкладки `Interface` і вибираємо в випадаючому меню «повзунок» (Slider). Наш повзунок регулює значення змінної `number` (рис. 4.12), яка буде змінюватися в діапазоні від 0 до 2000 з кроком 10.

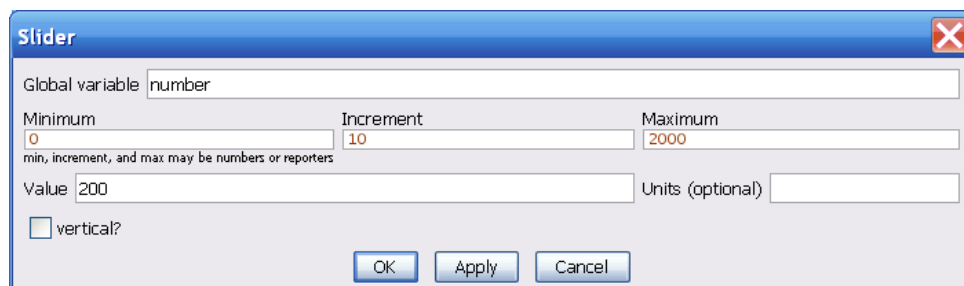


Рисунок 4.12

Тепер усе гаразд. Втім, ні. Навряд чи світлячкам властиво крутитися абсолютно безцільно, як це описано в `move`. Нехай наші світлячки здійснюють випадковий поворот праворуч або ліворуч, перед тим як продовжити рух вперед. Новий варіант `move` має вигляд

```
to move
  ask turtles [
    rt random-float 90 – random-float 90
    fd 1
  ]
End
```

В `go` ми повинні додати процедуру, яка збільшує значення лічильника `clock` на кожному кроці і змінює колір світлячка відповідно до значення `clock`.

```
to go
  move
  ask turtles [
    increment-clock
  ]
  ask turtles [
    ifelse (clock < threshold)
    [ set color yellow ]
    [ set color gray ]
  ]
End

to increment-clock
  set clock (clock + 1)
  if clock = 10
    [ set clock 0 ]
End
```

Ми використали нову команду `if (test) [commands]`, яка є спрощеним варіантом `ifelse`. Якщо повернутися до вкладки інтерфейс і запустити модель, натиснувши `go-one-step`, ми побачимо, що світлячки жовтого кольору (тобто мерехтливі) на наступному кроці гаснуть (стають сірими) (рис. 4.13).

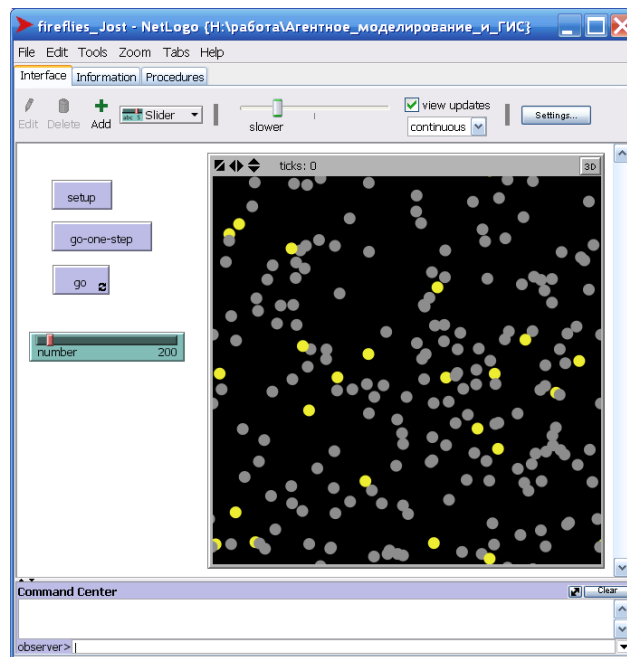


Рисунок 4.13

4. Додаємо правило синхронізації

Тепер наші світлячки рухаються по колу і мерехтять кожні 10 кроків. Однак в їх мерехтінні немає ніякої синхронності. Чому? Ну, хоча б тому, що ми не реалізували найпростішого правила такої синхронізації. Воно полягає в наступному: коли сусідній світлячок спалахує, то clock встановлюється у такому значенні, яке дорівнює threshold. Отже, ми потребуємо процедури look, щоб стежити за спалахами сусідніх світлячків.

to look

if (count turtles in-radius 1 with [color = yellow] >= 1)
[set clock reset-level]

End

Що робить цей тест? По-перше, він підраховує кількість мерехтливих світлячків по сусідству з даними, а по-друге – перевіряє чи більший він ніж одиниця (одже і сам світлячок знаходиться у власній околиці). Ця процедура викликається після increment-clock, якщо світлячок не мерехтить сам, і став знову чутливий до мерехтіння оточуючих його родичів.

to go

move

ask turtles [

increment-clock

if ((clock > window) and (clock >= threshold))

```

[ look ]
]
ask turtles [
  ifelse (clock < threshold)
  [ set color yellow ]
  [ set color gray ]
]
End

```

Ну ось, правило, яке синхронізує мерехтіння світлячків, готове. Перевіримо, чи працює воно.

5. Покращуємо інтерфейс

Наша модель працює. Тепер з її допомогою можна вивчати, як різні зовнішні чинники впливають на синхронізацію. Скільки часу, наприклад, займе синхронізація 80% світлячків? Як впливає на цей час чисельність популяції? А як впливає значення window? Експерименти з нашою моделлю допоможуть виявити нові закономірності в груповій поведінці світлячків, які потім можна буде перевірити в природних умовах.

Щоб зробити роботу з моделлю більш зручною, нам належить внести кілька поліпшень в інтерфейс. Щоб вивести на екран кількість кроків роботи моделі, створимо новий елемент інтерфейсу – датчик (monitor). Вибираємо на панелі інструментів в Interface пункт Monitor і вказуємо, що необхідно вивести значення ticks (рис.4. 14).

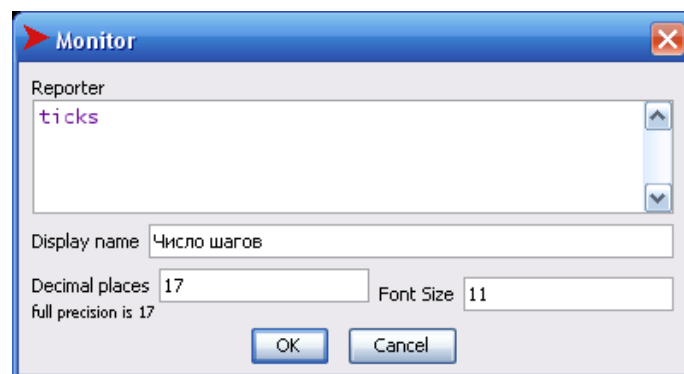


Рисунок 4.14

Ця вбудована команда повертає нам ту кількість кроків, які нас цікавлять. Розмістимо її також у процедуру go, щоб значення мало змогу поновлюватися на кожному кроці роботи.

На екрані з'являється датчик «Число кроків», значення якого дорівнює 0 після натискання `setup` і збільшується на одиницю після кожного натискання на `go-one-step`. Як можна побачити, ніщо не заважає використовувати для називання елементів інтерфейсу російську мову (рис. 4.15).

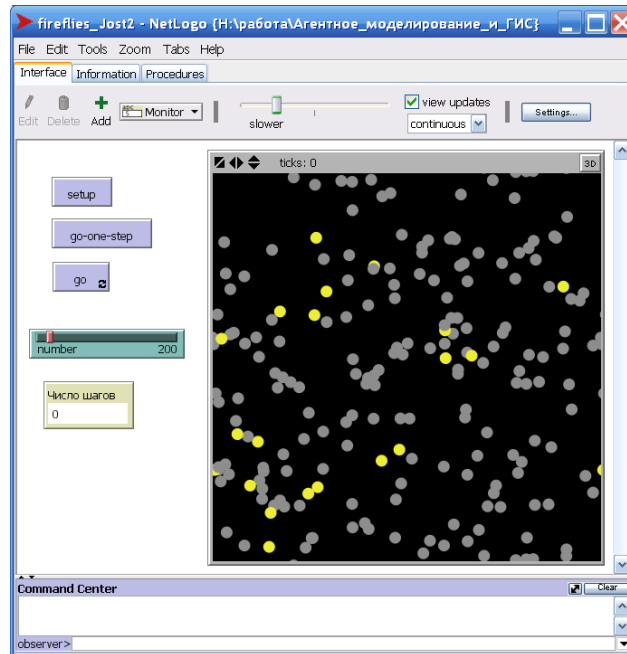


Рисунок 4.15

Добре б знати також кількість мерехтливих в даний момент світлячків. Для цього створимо датчик `Flashes`, заснований на глобальній змінній – `flashes`.

Глобальну змінну оголосимо, додавши рядок `globals [flashes]` після команди `turtles-own [...]`. Код процедури, який підраховує мерехтливих світлячків виглядає так:

```
to count-flashes
  set flashes ( count turtles with [color = yellow] )
end
```

`count-flashes` може викликатися в кінці процедури `go`, а як створити датчик ми вже знаємо. Тепер, виконуючи програму крок за кроком, ми зможемо визначити момент, коли синхронізація охопить понад 80% популяції світлячків. Спробуйте зробити це самостійно.

Довго? Згоден. Набагато зручніше було б виводити на екран графік, який зображує кількість мерехтливих світлячків (`flashes`) як функцію часу (`steps`). Природно, NetLogo дозволяє виводити графіки. Для цього потрібно створити процедуру, що задає параметри даного графіка, а потім – процедуру, яка оновлює

дані на графіку на кожному кроці виконання моделі. Отже, в процедуру setup додамо виклик процедури

```
create-plot
  to create-plot
    set-current-plot «Flashing Fireflies»
    set-plot-y-range 0 number
  end
```

і в go – виклик процедури do-plot

```
to do-plot
  set-current-plot «Flashing Fireflies»
  plot flashes
end
```

Процедура create-plot, розбираючи її роботу рядками, встановлює наступне:

- вікно для малювання графіка;
- діапазон значень по осі у: від 0 до number (ясно, що чисельність мерехтливих світлячків не може бути більше загальної);

Процедура do-plot вибирає графік «Flashing Fireflies» і додає до нього чергове значення flashes.

Щоб set-current-plot «Flashing Fireflies» спрацював, необхідно створити відповідний елемент інтерфейсу і змінити його ім'я на те, яке використовуються в коді (Flashing Fireflies). Крім того, тут ми можемо поставити автоматичне масштабування осі координат (Autoplot), і колір кривої, виведеної на графіку, (в нашому випадку: червоний) (рис. 4.16).

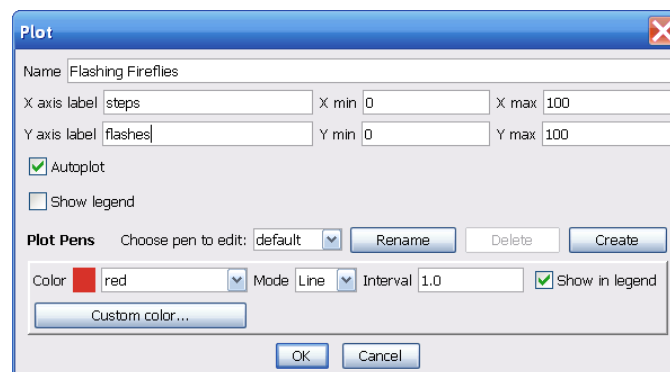


Рисунок 4.16

Наведемо тепер порядок в розташуванні елементів інтерфейсу. Для цього, клацнемо правою кнопкою миші той елемент, який нас цікавить. Вибираючи *Select*, можна змінювати розміри або розташування елемента. Тепер натискаємо *setup*, потім *go* (рис.4.17).

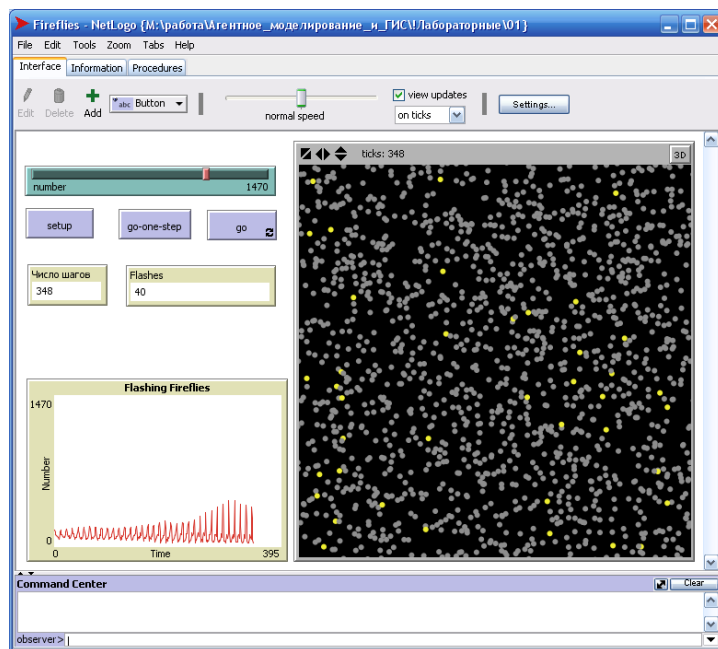


Рисунок 4.17

Отже, в наших руках найпростіша модель для дослідження завдання синхронізації мерехтіння світлячків. Звичайно, процедури можна було б написати і краще (в *File / Models Library*, в розділі *Biology*, є більш акуратна реалізація цієї моделі). Проте, модель працює, а далі ви самі можете її розвивати і вивчати поведінку світлячків (код даної моделі знаходиться в файлі моделі / *Fireflies.nlogo*).

Вивчіть, як чисельність світлячків впливає на час синхронізації 80% їх популяції (встановлюйте *number* рівним 10, 20, 50, 100, 200, 500, 1000, і пробуйте). Переконайтеся в тому, що отриманий результат стійкий, запускаючи модель кілька разів з одними і тими ж параметрами.

Розгляньте, як для фіксованої чисельності світлячків (наприклад, 500), цей інтервал залежить від значення змінної *window* (часу, протягом якого світлячок не звертає увагу на мерехтіння сусідів). Додайте повзунок або вікно введення для зміни значення *window*. Як можна трактувати цей ефект з точки зору біології?

Запропонований спосіб синхронізації мерехтінь далеко не єдиний. Чи можете ви поліпшити існуючий спосіб або запропонувати свій?

6. Інформація про модель

Детальна інформація про кожну модель знаходиться у вкладці Information (рис. 4.18).

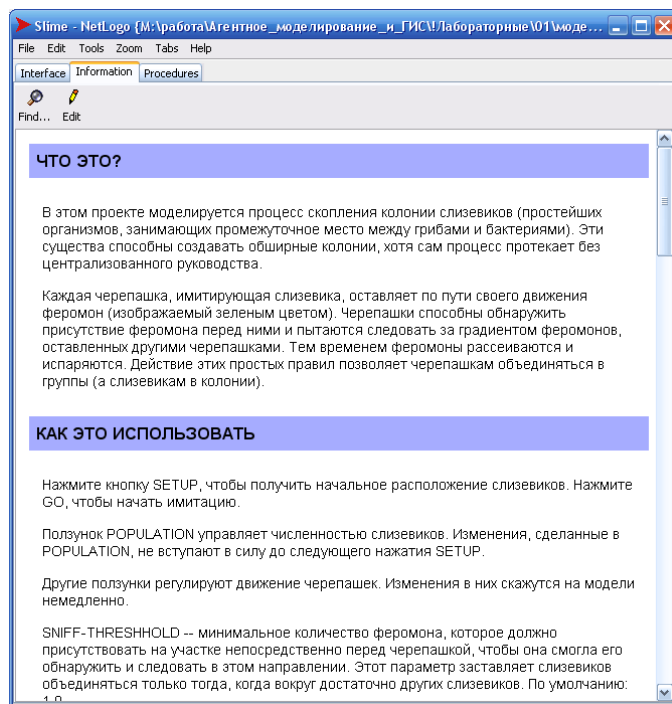


Рисунок 4.18

7. Індивідуальні завдання

Файли * .nlogo із завданнями знаходяться в папці *моделі*. Номер завдання відповідає вашому номеру в списку групи.

1. Терміти (Termites)
2. Банківські резерви (Bank Reserves)
3. Лісова пожежа (Fire)
4. Мурахи (Ants)
5. Епідемія (Virus)
6. Транспортний потік (Traffic Basic)
7. Поширення комп'ютерного вірусу (Virus on a Network)
8. Бджолині соти (Honeycomb)
9. Фільтрація води крізь ґрунт (Percolation)
10. Створення зграї (Flocking)
11. Система частинок (Particle System Basic)
12. Система типу «хижак-жертва» (Rabbits Grass Weeds)
13. Оптимізація методом рою частинок (Particle Swarm Optimization).

Контрольні питання за темою:
«Інформаційні системи імітаційного моделювання та конструювання»

1. Як запустити модель на виконання?
2. Що таке віртуальний час?
3. Як переключитися з режиму віртуального часу в реальний?
4. Поняття статистичного та імітаційного моделювання.
5. Основна перевага імітаційного моделювання.
6. Недоліки імітаційного моделювання.
7. Основні процедури імітаційного моделювання.
8. Які класи активних об'єктів включає проект?
9. Як змінити поточні значення змінних і параметрів моделі при її виконанні?
10. Як змінити швидкість виконання моделі?
11. У чому сенс експерименту в програмі AnyLogic?
12. Які типи експериментів підтримуються програмою AnyLogic?
13. Загальні властивості імітаційного моделювання?
14. Як ви розумієте детерміноване моделювання?
15. За яким принципом здійснюється просування модельного часу в імітаційній моделі.

Список джерел інформації

1. Информационные системы конструирования и моделирования объектов. Навчальн. посібник /. Адашевская И.Ю. /. Харків: «НТМТ», 2016.– 200 с.
2. Лычкина Н.Н. Имитационное моделирование экономических процессов / Н.Н. Лычкина, 2005.
3. Замятина О.М. Использование Advanced Process Panel и AdvancedTransferPanel в среде Arena 7.0 для моделирования и анализа сложных систем / О.М. Замятина, Н.Г. Саночкина .– Томск: Изд. ТПУ, 2005 (acs.cctpu.edu.ru/books.shtml).
4. Карпов Ю.Г. Имитационное моделирование систем / Ю.Г. Карпов, – 2005.
5. Варжапетян А.Г. Имитационное моделирование на GPSS / А.Г. Варжапетян. – учеб. пособ. – СПб.: ГУАП, 2007. – 384 с.
6. Kelton W.D., Sadowski R.P., Sadowski D.A. Simulation with Arena McGraw-Hill, Boston, 2012.– 547 p.
7. Замятина О.М. Использование Advanced Process Panel и AdvancedTransferPanel в среде Arena 7.0 для моделирования и анализа сложных систем / О.М. Замятина, Н.Г. Саночкина .– Томск: Изд. ТПУ, 2005 (acs.cctpu.edu.ru/books.shtml).
8. Arena Basic Edition User's Guide. Rockwell Software, 2009. – 82 с.

ДЛЯ ПРИМІТОК

ДЛЯ ПРИМІТОК

Навчальне видання

МЕТОДИЧНІ ВКАЗІВКИ

**Методи конструювання об'єктів в комп'ютерних системах
(Частина IV)**

для студентів спеціальності 122 «Комп'ютерні науки»

Укладач: АДАШЕВСЬКА Ірина Юріївна

За авторською редакцією

План 2018 р., поз. 70.

Підписано до друку 24.05.18 . Формат 60×84 1/16.

Папір друк. № 2.

Друк-ризографія. Гарнітура Times New Roman. Ум. друк. арк.

Наклад 50 прим. Зам. № 12. Ціна договірна.

Видавничий центр НТУ «ХПІ». 61002, Харків, вул. Кирпичова, 2.

Свідоцтво про реєстрацію ДК № 3657 від 24.12.2017 р.

Друкарня НТУ «ХПІ», 61002. Харків, вул. Кирпичова, 2